

Technical Report #5-20446
Contract Number DAAH01-98-R001
Delivery Order No. 37

**(U) 3Dpimms Verification and Blast Methodology Study
(5-20446)**

Final Technical Report

February 2000

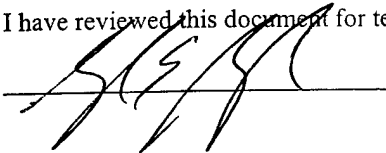
Prepared by:

*Glenn E. Romanczuk
Chris Pitts
Will Lee*

Visualization & Simulation Laboratory
Research Institute
The University of Alabama in Huntsville
Huntsville, Alabama 35899

Prepared for
Aeroballistics Analysis Functional Area
Research, Development, and Engineering Center
U.S. Army Aviation & Missile Command
Redstone Arsenal, Alabama 35898
Attn: Ms Edith Crow AMSAM-RD-SS-AA

I have reviewed this document for technical and security purposes and find it acceptable.

A handwritten signature in black ink, appearing to be 'S. E. R.', is written over a horizontal line.

20000502 111

REPORT DOCUMENTATION PAGE**Form Approved**
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing this collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503

1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE January, 2000	3. REPORT TYPE AND DATES COVERED Final, 1/21/99 - 12/31/99	
4. TITLE AND SUBTITLE 3DPimms Verification and Blast Methodology Study			5. FUNDING NUMBERS	
6. AUTHOR(S) Mr. Glenn E. Romanczuk, Mr. Chris Pitts				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) The University of Alabama in Huntsville, Research Institute 301 Sparkman Drive, RI E-47 Huntsville, Alabama, 35899			8. PERFORMING ORGANIZATION REPORT NUMBER 5-20446	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) U.S. Army Aviation & Missile Command AMSAM-RD-SS-AA Commander, AMCOM AMSAM-RD-SS-AA Redstone Arsenal, AL 35898			10. SPONSORING / MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES				
12a. DISTRIBUTION / AVAILABILITY STATEMENT UnClassified/Unlimited			12b. DISTRIBUTION CODE A	
13. ABSTRACT (Maximum 200 Words) This report documents the efforts of the University of Alabama in Huntsville to assist in the Verification and Validation of a new MOUT lethality code 3DPimms. This code has been constructed over the past three years to replace PIMMS. A Verification and Validation was authored and approved by the accrediting agency, EAC (now AEC) and a Verification and Validation draft report was authored and is awaiting approval. A large amount of uncertainty exists in the next items for this methodology to include. Therefore, plans include contingencies for the inclusion of Blast and other insults which can cause incapacitation. A unique method of using a blast transport code and Blast code derived from a mechanical model of chest wall/lung interaction has been postulated. This method is recommended as a first cut attempt to include blast in MOUT analysis before ORCA is available. A tool to display and analyze pk maps was also generated during this work. This tool allows for simulation data to be merged with Pk maps of interest and display an overall answer for system lethality.				
14. SUBJECT TERMS Verification, Lethality, Visualization, Simulation, ORCA, Effectiveness, Incapacitation			15. NUMBER OF PAGES 87	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT Unlimited	

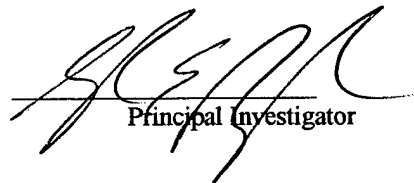
PREFACE

- (U) This technical report was prepared by the staff of the Visualization & Simulation Laboratory of the Research Institute, The University of Alabama in Huntsville. It documents the research performed under contract number DAAH01-98-R001, delivery Order 0037. Mr. Glenn E. Romanczuk served as the Principal Investigator Ms. Edith Crow of the AMCOM Aeroballistics Analysis functional area provided the technical coordination..
- (U) The views, opinions, and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision unless so designated by other official documentation.
- (U) Except as provided by the Contract Data Requirements List DD 1432, whereif, the distribution of any contract report in any stage of development or completion is prohibited without the approval of the Contracting Officer.

Prepared for:

Commander
U.S. Army Aviation & Missile Command
Redstone Arsenal, Alabama 35898

- (U) I have reviewed this report , dated February, 2000 and the report is unclassified.



Principal Investigator

LIST OF FIGURES

Figure 1 & 2 , The Qk window and the main Menu modification

Figure 3 - The TLT in Windows

Figure 4 - The Config Dialog in Windows

Figure 5 – An example of the Didi weight code

Figure 6 - A test image showing spall paths

Figure 7 - PiMan on the PC

Figure 8 - The PIMan code on the Silicon Graphics

Figure 9 - Simvu displaying bunker testfile

Figure 10 - Simvu displaying APC test image

Figure 11- The Interactive Analysis environment in ProspectV2

Table of Contents

INTRODUCTION	5
SCOPE OF WORK	5
RESULTS	5
MAINTAIN SOFTWARE	5
WEIGHTING CODES	7
LETHALITY WORKING GROUP MEETINGS	7
BLAST AND THERMAL METHODOLOGY	8
<i>Blast Research</i>	8
<i>Methodology creation and review with ARL</i>	8
V&V EFFORTS	9
GENERAL LETHALITY SUPPORT	9
<i>APC Lethality Visualization</i>	9
<i>PiMan Tools</i>	10
<i>Simulation Results and Target Viewers</i>	10
<i>General geometric linkages to wound ballistics models</i>	11
<i>Virtual Reality and V&V Tools</i>	16
CONCLUSIONS	20
APPENDIX I - V&V PLAN	21
APPENDIX II - V&V DRAFT REPORT	65
APPENDIX III - GEOMETRY POSSIBILITIES	82

Introduction

This contract between the University of Alabama in Huntsville, Research Institute and the Systems Simulation and Development directorate of the U.S. Army Aviation & Missile Command covered the specific items and engineering services which are presented in the scope of work section of this report.

The UAH reference number for this work is Account number 5-20446 and is entitled F/DOD/ARMY/AMCOM/LETHALITY SUPPORT FOR MPIM. The period of performance was 1/21/99 to 12/31/99.

Scope of Work

The following items are listed in the scope of work for this task order contract with U.S. Army Aviation & Missile Command.

1. Maintain current software, NEWP, as system upgrades are required.
2. Participate in quarterly Lethality Working Group Meetings.
3. Provide software support to upgrade GUI and formulate plans to include blast and thermal data into the methodology.
4. Upgrade to the latest version of ComputerMan.
5. Provide software support in the area of implementation and use of codes developed. Investigate ORCA and provide hooks for implementation.
6. Assist with V&V efforts on 3Dpimms and coordinate with AMSAA and EAC.

Results

Maintain Software

The current version of 3Dpimms was maintained and several new capabilities were added. The first capability was the addition of a button to allow the analyst access to a special form of Probability of Kill file. These files have a special format for vehicles with a specific number of personnel inside. However, the program was written to handle any number of data columns.

The program is a variant of the standard PK code which UAH has utilized for the past 10 years. However, this code is customized to work with 3Dpimms and special files which group Probability of incapacitation and aggregate it in a specific manner. The user is cautioned to remember this fact.

Figure 1 contains an image of the qk program and Figure 2 shows the place that the analyst pushes to generate the appropriate graphical output.

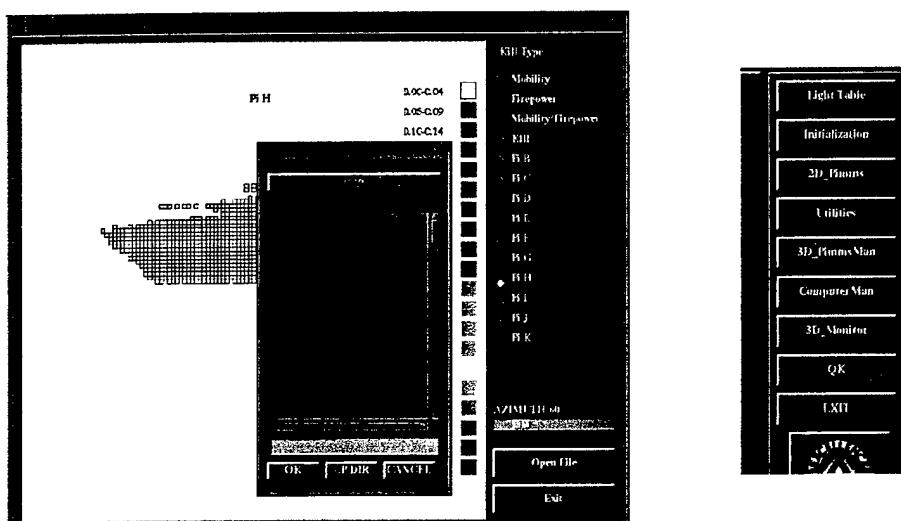


Figure 1 & 2 , The Qk window and the main Menu modification

Additionally, files were installed on the AMSAA Octane for AMSAA personnel to utilize. Also, under development is a standard release version which will allow for all directories and environment variables to be standardized. This update and layout of the proposed directory tree will be checked by the customer and then delivered to AMSAA. Significant efforts were also expended to begin the exploration of using the Windows environment in addition to Unix. Several tools were prototyped which would allow an analyst to use Windows for an entire analysis. Figure 3 & 4 show these working prototypes.

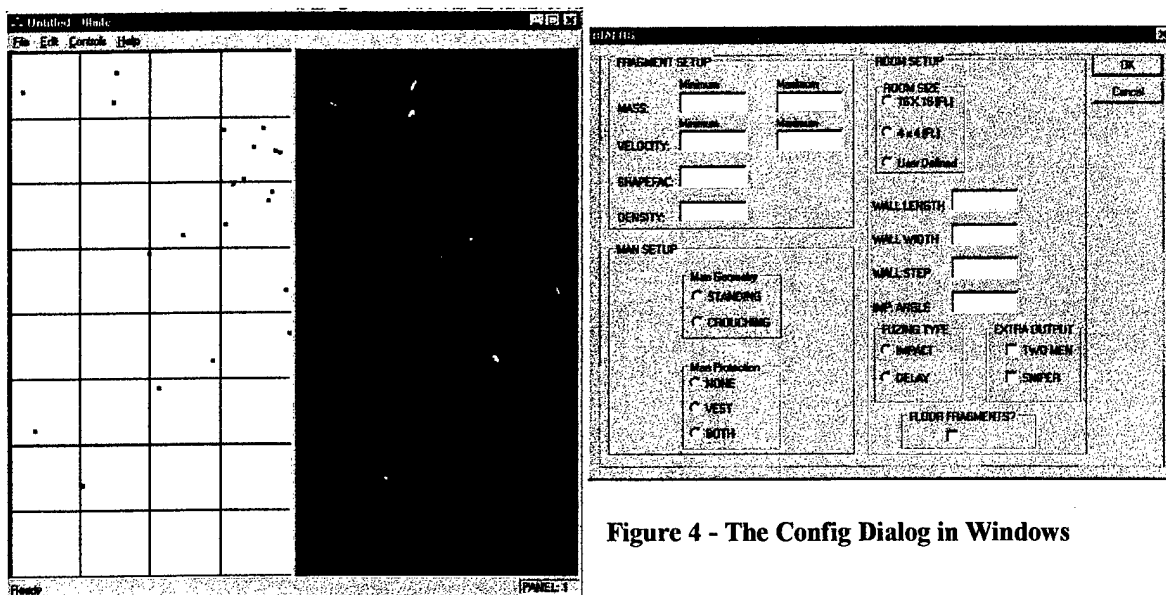


Figure 3 - The TLT in Windows

Figure 4 - The Config Dialog in Windows

Weighting codes

Several weighting codes were written from customer requirements. Simulation data from a 6DOF of the MPIM system is available and this information must be tied together with the Pk data from the vehicles of interest and the MOUT targets. Figure 5 shows a sample image from this code.

Lethality Working Group Meetings

Numerous Lethality Working Groups and quarterly IPR's were attended during the course of this contract. Several critical and key decisions were reached during these meetings. The list of key areas reached is too numerous to detail in this report but a major result, which was achieved due to UAH assistance, was the final definition of a squad kill or incapacitation. This method includes the definition of a new performance indicator the merged Probability of Mobility or Firepower or Incapacitation. (Pkmfi).

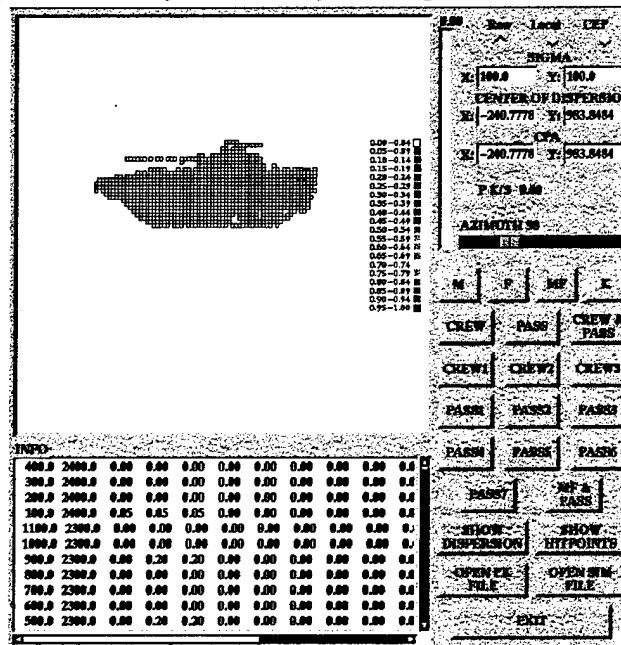


Figure 5 – An example of the Didi weight code.

The above mentioned agreement and definition ends three years of effort to define a method of calculation of a weighted Measure of Effectiveness (MOE) that meets the intent of the MPIM Requirements documents.

Additionally, EAC has reached an agreement to accredit 3Dpimms methodology as the Evaluation tool for the MPIM Evaluation. Please refer to the V&V section of this report for more information.

Blast and Thermal methodology

Blast Research

Several key documents were gathered and analyzed for creation of a prototype inclusion of blast data into 3Dpimms. This work traced the available blast data and analysis from the 1950's to current methods using mechanical models of the chest / lung system. The results are too numerous to place in this brief report, however, major findings can be summarized in the following sections.

Several missile systems at Redstone are undergoing tests which allow a blast environment to be collected and evaluated. Working with our COTR, inputs to the methodology and subsequent report were generated. The report authored by Brian Smith and Edith Crow, is entitled, "Personnel Lethality Prediction Techniques for Military Operations in Urban Terrain (U) , TR-RD-SS-00-XX. The definitions and the tools and methodology in the next section are a quick review of the major findings.

Methodology creation and review with ARL

Mr. Dave Neades serves in many capacities within the Army Research Laboratory. One of his current projects (ORCA) involves the creation of a many insult on one person incapacitation and casualty model. This model incorporates many existing legacy models for insults to people. The two that are of current interest are the INJURY model from the Walter Reed Army Research Institute and the COMPUTERMAN wound ballistics model from ARL.

3Dpimms is in the process of attempting to utilize the results of Mr. Neades' efforts by replacing our current implementation of ComputerMan with the tool ORCA. However, in order to use the blast module, a method had to be created to use test data to determine the blast environment at any place a man may be located in the standard bunker or "16 x 16 room." A tool was identified through Mr. Matt Rosenblatt of AMSAA which allows blast energy to be calculated from physics based models and output the desired Pressure vs. Time history at given points from a Blast within and enclosure. This code is from the Waterways Experiment Stations and is called BLASTX. The code is simple and modular. It runs on a standard Personal Computer under the Windows operation system. It has a batch mode capability once set up which allows large runs to be completed quickly. This code provides an accepted physics based approach to allow blast data from test to calibrate a room by modifying the charge inputs until there is agreement with the test data. Once this characterization step is completed the Pressure vs. Time histories for all man locations can be computed.

At this point it should be noted that the BEAMS methodology developed by ARA, should also be considered. This methodology utilizes BRL-CAD raytrace and room or target files and propagates blast energy from the blast source to other locations in the target file. It is likely that the same type of Pressure vs. Time history needed could be

gathered from this approach. This will be important if these methods become standard within the MUVES/AJEM codes. The methodology concept with P vs. T curves per man location and burst point would be to allow ORCA to determine incapacitation. However, early alpha tests showed a lack of correlation to entered test data.

Due to the unknown alpha test failures, an alternate methodology was proposed and accepted. The previous mentioned method would be used to generate the pressure time histories, however, the standalone INJURY model would be utilized from Walter Reed. This approach will create a matrix that will allow blast insults to be evaluated first in the 3Dpimms method. This will aide run time as the people nearest to the burst point should be incapacitated due to blast. These are the very people whom are usually hit with the most fragments and add to the run time of the high resolution methodology. This approach was agreed upon at the last LWG meeting and should be created using FY2000 money.

V&V efforts

The V&V efforts can be broken down into two sections. The first is the planning document which was required by AEC (Formerly, EAC). This document can be found in Appendix I and details the process which would establish the Verification and Validation of the 3Dpimms methodology. The second area was the actual report developed as a part of the V&V effort. This document is found in draft form in Appendix II. The draft form is necessary because this document will be bound with the COTR's document on the validation of the method. This document also can serve as a beginning ASPI document should JTCG/ME accreditation be needed or conversion to HLA.

General Lethality Support

APC Lethality Visualization

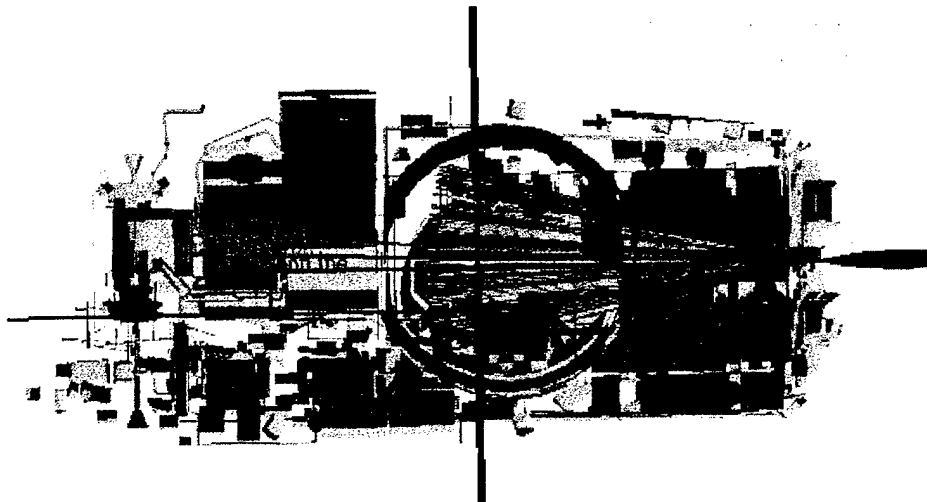


Figure 6 - A test image showing spall paths

Figure 6 shows one result from a tool which allows for a number of rays to be traced within a cone angle once the interior of a vehicle is reached. This tool builds on other work and allows for any penetration routine to be hooked to each ray that hits part of the target. In our case, the personnel in the vehicle are the critical components which will trigger more calculation. In the next sections, the development of modules which can be used for this purpose are explained.

PiMan Tools

Several tools were written to develop the modules and methodologies for comparison of personnel incapacitation models. The tools created on the PC and on the SGI allow a view of the crouching or standing man to be seen. Using the mouse, the user selects the ray and shoots the specific fragment. In the background a BRL-CAD raytrace is started and the appropriate body part and entry and exit position is returned. Standard coding of the Sperrazza-Kokinakis and the Ballistic Dose method is used to fill the output screen with the results from those incapacitation methodologies. Figure 7 shows the interface as it looks in the Windows environment.

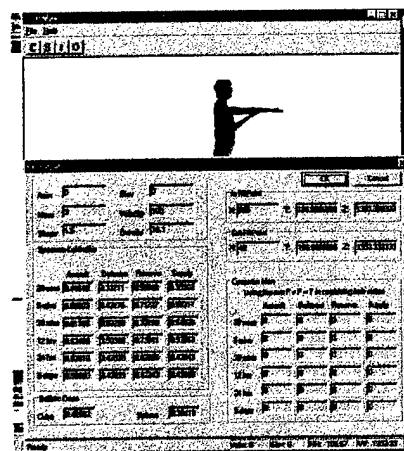


Figure 7 - PiMan on the PC

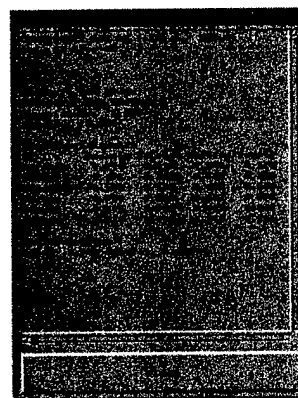
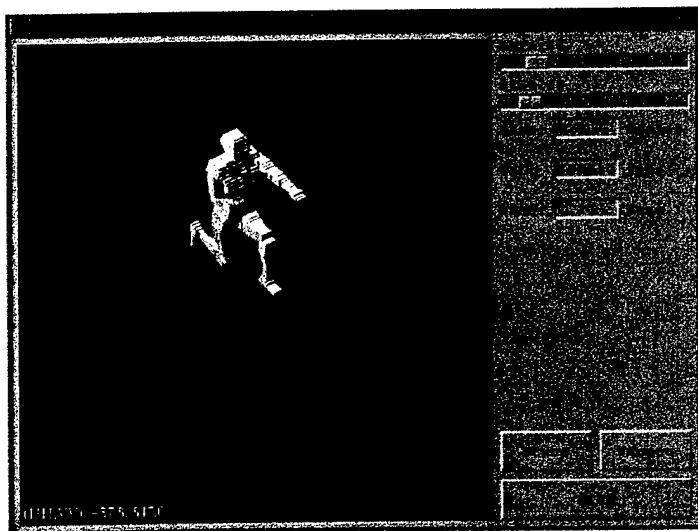


Figure 8 - The PiMan code on the Silicon Graphics

Simulation Results and Target Viewers

Simvu was written to show the target and dispersion data on line drawings of targets at all aspects. The main area of emphasis was the creation of the line drawings in both tiff and vector form. Figure 9 shows the basic layout with toggles for the various vehicles and a file select box to read from the simulation file of interest.

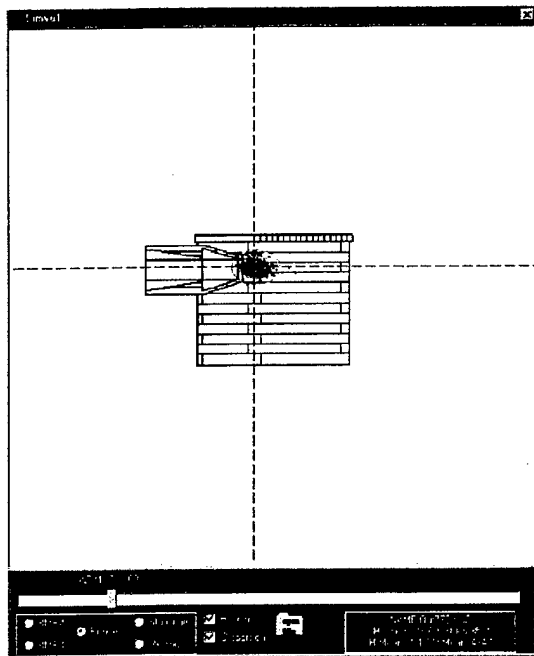


Figure 9 - Simvu displaying bunker testfile

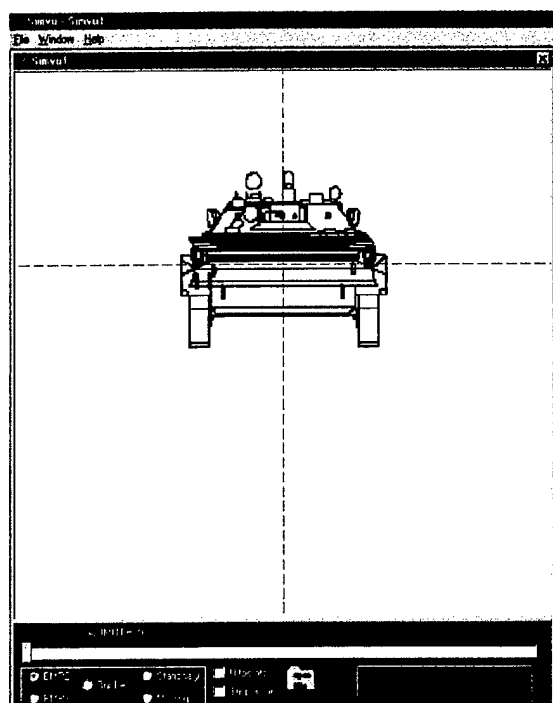


Figure 10 - Simvu displaying APC test image

General geometric linkages to wound ballistics models

See appendix III

This is a sample of the types of programs that were written to attempt to create alternate forms of men to use in 3Dpimms. It should not be utilized without further testing and evaluation.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define CELL_WIDTH 5.0
#define CELL_DEPTH 5.0
#define N_COLS 107
#define N_ROWS 50
#define N_SECS 167

#define LEFT_UPR_ARM 0
#define LEFT_LWR_ARM 1
#define RIGHT_UPR_ARM 2
#define RIGHT_LWR_ARM 3
#define LEFT_UPR_LEG 4
#define LEFT_LWR_LEG 5
#define RIGHT_UPR_LEG 6
#define RIGHT_LWR_LEG 7
#define HEAD_NECK 8
#define TORSO 9
#define THORAX 10
#define ABDOMEN 11
#define PELVIS 12

const int startsec[13] = {50, 59, 109, 118, 76, 91, 135, 150, 1, 1, 19, 28, 36};
const int stopsec[13] = {58, 75, 117, 134, 90, 108, 149, 167, 18, 44, 27, 35, 44};

const int lowerlimit[8] = {-80, 0, -80, 0, -30, -110, -30, -110};
const int upperlimit[8] = {180, 155, 180, 155, 110, 0, 110, 0};

const int MINROW[1+N_SECS] = {
```

```

1,
15, 12, 11, 9, 8, 9, 9, 8, 8, 6,
5, 8, 9, 9, 8, 8, 11, 15, 18, 18,
16, 15, 13, 9, 6, 4, 4, 2, 1, 1,
1, 1, 1, 2, 3, 3, 4, 6, 8, 11,
14, 15, 14, 19, 51, 51, 51, 51, 51, 21,
21, 21, 21, 20, 20, 20, 21, 22, 23, 23,
23, 22, 21, 22, 22, 23, 24, 24, 24, 25,
25, 24, 24, 27, 28, 19, 20, 20, 21, 21,
21, 22, 23, 23, 23, 23, 23, 22, 21, 20,
21, 22, 23, 24, 24, 25, 25, 26, 26, 26,
26, 26, 26, 26, 25, 25, 22, 1, 21, 21,
21, 21, 20, 20, 20, 21, 22, 23, 23, 23,
22, 21, 22, 22, 23, 24, 24, 24, 25, 25,
24, 24, 27, 28, 19, 20, 20, 21, 21, 21,
22, 23, 23, 23, 23, 22, 21, 20, 21,
22, 23, 24, 24, 25, 25, 26, 26, 26, 26,
26, 26, 26, 25, 25, 22, 1
};

const int MAXROW[1+N_SECS] = {
0,
37, 40, 42, 43, 44, 46, 46, 47, 46, 45,
44, 43, 42, 41, 41, 41, 40, 38, 42, 46,
46, 47, 47, 47, 46, 46, 46, 45, 45, 44,
45, 45, 45, 44, 43, 44, 45, 46, 47, 48,
48, 49, 50, 49, 50, 50, 50, 50, 50, 43,
41, 41, 40, 39, 38, 37, 37, 37, 36, 36,
36, 36, 35, 35, 35, 34, 34, 33, 32, 32,
33, 33, 33, 33, 33, 49, 49, 49, 49, 49,
49, 48, 48, 46, 46, 44, 44, 43, 43, 42,
42, 42, 43, 43, 43, 43, 43, 41, 40, 40,
39, 39, 39, 39, 39, 41, 44, 46, 43, 41,
41, 40, 39, 38, 37, 37, 37, 36, 36, 36,
36, 35, 35, 35, 34, 34, 33, 32, 32, 33,
33, 33, 33, 33, 49, 49, 49, 49, 49, 49,
48, 48, 46, 46, 44, 44, 43, 43, 42, 42,
42, 43, 43, 43, 43, 43, 41, 40, 40, 39,
39, 39, 39, 39, 41, 44, 46
};

const int MINCOL[1+N_SECS] = {
1,
45, 42, 40, 40, 39, 39, 38, 36, 37, 37,
37, 39, 40, 40, 41, 42, 42, 42, 31, 15,
13, 11, 10, 21, 22, 24, 25, 24, 25, 25,
25, 26, 27, 27, 28, 27, 26, 26, 26, 24,
24, 23, 23, 23, 1, 1, 1, 1, 1, 9,
7, 7, 7, 6, 5, 5, 5, 5, 4, 4,
4, 4, 4, 5, 5, 5, 5, 6, 5, 5,
5, 3, 1, 3, 3, 21, 21, 21, 21, 23,
23, 24, 25, 26, 28, 29, 29, 30, 30, 29,
30, 31, 31, 31, 31, 32, 32, 33, 33, 34,
34, 35, 36, 36, 35, 33, 33, 32, 88, 87,
85, 84, 85, 86, 86, 86, 85, 84, 85, 85,
86, 86, 87, 88, 90, 90, 90, 91, 91, 89,
88, 88, 89, 90, 55, 56, 57, 57, 58, 58,
58, 58, 59, 59, 59, 59, 59, 59, 59, 59,
59, 59, 59, 59, 59, 60, 60, 62, 63, 63,
64, 64, 64, 63, 63, 61, 60
};

const int MAXCOL[1+N_SECS] = {
0,
63, 66, 67, 68, 69, 69, 69, 70, 70, 70,
69, 68, 68, 67, 66, 66, 66, 65, 77, 92,
94, 96, 98, 86, 85, 84, 83, 83, 82, 82,
82, 82, 81, 81, 80, 81, 81, 81, 82, 83,
84, 84, 84, 85, 0, 0, 0, 0, 0, 20,
21, 23, 24, 23, 22, 22, 22, 23, 24, 23,
23, 22, 22, 21, 20, 18, 18, 18, 17, 17,
19, 20, 20, 19, 18, 53, 52, 51, 51, 50,
50, 50, 50, 49, 49, 49, 49, 49, 49, 49,
49, 49, 49, 49, 49, 49, 48, 48, 46, 45,
45, 44, 44, 44, 45, 45, 47, 48, 99, 101,
101, 101, 102, 103, 103, 103, 103, 104, 104, 104,
104, 104, 103, 103, 103, 103, 102, 103, 103, 103,
105, 107, 105, 105, 87, 87, 87, 87, 85, 85,
84, 83, 82, 80, 79, 79, 78, 78, 79, 78,
77, 77, 77, 77, 76, 76, 75, 75, 74, 74,
73, 72, 72, 73, 75, 75, 76
};

const int ZMIN[1+N_SECS] = {
0,
1738, 1726, 1714, 1702, 1690, 1678, 1666, 1654, 1642, 1630,
1618, 1606, 1594, 1582, 1570, 1558, 1546, 1534, 1508, 1482,
1456, 1430, 1404, 1378, 1352, 1326, 1300, 1274, 1248, 1222,
1196, 1170, 1144, 1118, 1092, 1066, 1040, 1014, 988, 962,
936, 910, 884, 858, 0, 0, 0, 0, 0, 1378,

```

```

1352, 1326, 1300, 1274, 1248, 1222, 1196, 1170, 1144, 1118,
1092, 1066, 1040, 1014, 988, 962, 936, 910, 884, 858,
832, 806, 780, 754, 728, 832, 806, 780, 754, 728,
702, 676, 650, 624, 598, 572, 546, 520, 494, 468,
442, 416, 390, 364, 338, 312, 286, 260, 234, 208,
182, 156, 130, 104, 78, 52, 26, 0, 1378, 1352,
1326, 1300, 1274, 1248, 1222, 1196, 1170, 1144, 1118, 1092,
1066, 1040, 1014, 988, 962, 936, 910, 884, 858, 832,
806, 780, 754, 728, 832, 806, 780, 754, 728, 702,
676, 650, 624, 598, 572, 546, 520, 494, 468, 442,
416, 390, 364, 338, 312, 286, 260, 234, 208, 182,
156, 130, 104, 78, 52, 26, 0
};

const int ZMAX[1+N_SECS] = {
0,
1750, 1738, 1726, 1714, 1702, 1690, 1678, 1666, 1654, 1642,
1630, 1618, 1606, 1594, 1582, 1570, 1558, 1546, 1534, 1508,
1482, 1456, 1430, 1404, 1378, 1352, 1326, 1300, 1274, 1248,
1222, 1196, 1170, 1144, 1118, 1092, 1066, 1040, 1014, 988,
962, 936, 910, 884, 0, 0, 0, 0, 1404,
1378, 1352, 1326, 1300, 1274, 1248, 1222, 1196, 1170, 1144,
1118, 1092, 1066, 1040, 1014, 988, 962, 936, 910, 884,
858, 832, 806, 780, 754, 858, 832, 806, 780, 754,
728, 702, 676, 650, 624, 598, 572, 546, 520, 494,
468, 442, 416, 390, 364, 338, 312, 286, 260, 234,
208, 182, 156, 130, 104, 78, 52, 26, 1404, 1378,
1352, 1326, 1300, 1274, 1248, 1222, 1196, 1170, 1144, 1118,
1092, 1066, 1040, 1014, 988, 962, 936, 910, 884, 858,
832, 806, 780, 754, 858, 832, 806, 780, 754, 728,
702, 676, 650, 624, 598, 572, 546, 520, 494, 468,
442, 416, 390, 364, 338, 312, 286, 260, 234, 208,
182, 156, 130, 104, 78, 52, 26
};

void articulate (int Bodypart, int Degrees, float v[4], float h[4], float A[4], float B[4], int flag);
void matrixMult (float (*m1)[4][4], float (*m2)[4][4], float (*result)[4][4]);
void matrixMult2 (float (*m1)[4][4], float (*colvect)[4], float (*result)[4]);

float Man[200][8][4], Man2[200][8][4];
int currentangle[8];

void main(int argc, char **argv)
{
    int i,j,k,Bodypart;
    float xmin,xmax,ymin,ymax,zmin,zmax;
    int currentangle2[8];
    float magA,magB;
    float vertex[4],vecA[4],vecB[4],height[4];
    FILE *fp;
    char line[256];

    if(argc < 2) {
        fprintf(stderr,"Usage: CreateTgcMan <posture file> \n");
        exit(0);
    }

    if((fp = fopen(argv[1],"r")) == NULL) {
        fprintf(stderr,"Could not open %s\n",argv[1]);
        exit(0);
    }

    for(i=0;i<8;i++) {
        if(!feof(fp)) {
            fprintf(stderr,"Error reading %s\n",argv[1]);
            fclose(fp);
            exit(0);
        }
        fgets(line,256,fp);
        sscanf(line,"%d",&currentangle2[i]);
    }

    fclose(fp);

    for(i=0;i<8;i++)
        currentangle[i] = 0;

    for (i=1; i<=167; i++)
    {
        Man[i][0][0] = (MINCOL[i] - 1) * CELL_WIDTH;
        Man[i][0][1] = (50 - MAXROW[i]) * CELL_DEPTH;
        Man[i][0][2] = ZMAX[i];
        Man[i][0][3] = 1.0;

        Man[i][1][0] = (MAXCOL[i]) * CELL_WIDTH;
        Man[i][1][1] = Man[i][0][1];
        Man[i][1][2] = ZMAX[i];
        Man[i][1][3] = 1.0;

        Man[i][2][0] = Man[i][1][0];
        Man[i][2][1] = (50 - MINROW[i] + 1) * CELL_DEPTH;
        Man[i][2][2] = ZMAX[i];
        Man[i][2][3] = 1.0;

        Man[i][3][0] = Man[i][0][0];
        Man[i][3][1] = Man[i][2][1];
        Man[i][3][2] = ZMAX[i];
        Man[i][3][3] = 1.0;
    }
}

```

```

Man[i][4][0] = Man[i][0][0];
Man[i][4][1] = Man[i][0][1];
Man[i][4][2] = ZMIN[i];
Man[i][4][3] = 1.0;

Man[i][5][0] = Man[i][1][0];
Man[i][5][1] = Man[i][1][1];
Man[i][5][2] = ZMIN[i];
Man[i][5][3] = 1.0;

Man[i][6][0] = Man[i][2][0];
Man[i][6][1] = Man[i][2][1];
Man[i][6][2] = ZMIN[i];
Man[i][6][3] = 1.0;

Man[i][7][0] = Man[i][3][0];
Man[i][7][1] = Man[i][3][1];
Man[i][7][2] = ZMIN[i];
Man[i][7][3] = 1.0;
}

for(i=1;i<=167;i++){

    if(i==45 || i==46 || i==47 || i==48 || i==49) continue;

    for(k=0;k<13;k++){
        if(i >= startsec[k] && i <= stopsec[k] ) {
            Bodypart = k;
            break; } }

    /*----- CENTER VERTEX -----*/
    vertex[0] = (Man[i][0][0] + Man[i][1][0]) / 2.0;
    vertex[1] = (Man[i][0][1] + Man[i][3][1]) / 2.0;
    vertex[2] = Man[i][0][2];
    vertex[3] = 1.0;

    /*----- HEIGHT -----*/
    height[0] = (Man[i][0][0] + Man[i][1][0]) / 2.0;
    height[1] = (Man[i][0][1] + Man[i][3][1]) / 2.0;
    height[2] = Man[i][4][2];
    height[3] = 1.0;

    /*----- VECTOR A -----*/
    vecA[0] = Man[i][1][0];
    vecA[1] = (Man[i][0][1] + Man[i][3][1]) / 2.0;
    vecA[2] = Man[i][0][2];
    vecA[3] = 1.0;

    /*----- VECTOR B -----*/
    vecB[0] = (Man[i][0][0] + Man[i][1][0]) / 2.0;
    vecB[1] = Man[i][0][1];
    vecB[2] = Man[i][0][2];
    vecB[3] = 1.0;

    if(Bodypart < 8){
        if(Bodypart == LEFT_LWR_ARM) {
            articulate (i, currentangle2[LEFT_UPR_ARM],vertex,height,vecA,vecB,LEFT_UPR_ARM);
            articulate (i, currentangle2[LEFT_LWR_ARM],vertex,height,vecA,vecB,-1); }
        else if(Bodypart == RIGHT_LWR_ARM) {
            articulate (i, currentangle2[RIGHT_UPR_ARM],vertex,height,vecA,vecB,RIGHT_UPR_ARM);
            articulate (i, currentangle2[RIGHT_LWR_ARM],vertex,height,vecA,vecB,-1); }
        else if(Bodypart == LEFT_LWR_LEG) {
            articulate (i, currentangle2[LEFT_UPR_LEG],vertex,height,vecA,vecB,LEFT_UPR_LEG);
            articulate (i, currentangle2[LEFT_LWR_LEG],vertex,height,vecA,vecB,-1); }
        else if(Bodypart == RIGHT_LWR_LEG) {
            articulate (i, currentangle2[RIGHT_UPR_LEG],vertex,height,vecA,vecB,RIGHT_UPR_LEG);
            articulate (i, currentangle2[RIGHT_LWR_LEG],vertex,height,vecA,vecB,-1); }
        else {
            articulate (i, currentangle2[Bodypart],vertex,height,vecA,vecB,-1); } }

    height[0] -= vertex[0];
    height[1] -= vertex[1];
    height[2] -= vertex[2];

    vecA[0] -= vertex[0];
    vecA[1] -= vertex[1];
    vecA[2] -= vertex[2];

    vecB[0] -= vertex[0];
    vecB[1] -= vertex[1];
    vecB[2] -= vertex[2];

    magA = sqrt(vecA[0]*vecA[0] + vecA[1]*vecA[1] + vecA[2]*vecA[2]);
    magB = sqrt(vecB[0]*vecB[0] + vecB[1]*vecB[1] + vecB[2]*vecB[2]);

    /*
    printf("in sec%d tgc %f %f %f %f %f %f %f %f %f %f %f\n",i,vertex[0],vertex[1],vertex[2],height[0],height[1],height[2],vecA[0],vecA[1],vecA[2],vecB[0],vecB[1],vecB[2],magA,magB);
    */
}

void articulate (int Bodypart, int Degrees,float v[4], float h[4],float A[4],float B[4],int flag)
{

```

```

static float xlate[4][4], rotat[4][4];
static float tr[4][4], trt[4][4];
float angle;
int start, stop, i, j, k, k2;
float vt[4], vt2[4];
float ht[4], ht2[4];
float At[4], At2[4];
float Bt[4], Bt2[4];

for(i=0; i<4; i++){
    vt[i] = v[i];
    ht[i] = h[i];
    At[i] = A[i];
    Bt[i] = B[i]; }

k2 = Bodypart;
for(k=0; k<13; k++){
    if(Bodypart >= startsec[k] && Bodypart <= stopsec[k] ) {
        Bodypart = k;
        break; } }

if(flag >= 0)
    start = startsec[flag];
else
    start = startsec[Bodypart];

angle = (float)(Degrees - currentangle[Bodypart]) * 3.1415927 / 180.0;
/*currentangle[Bodypart] = Degrees;*/

switch (Bodypart) /* For upper limbs, articulate lower limb with it */
{
    case LEFT_UPR_ARM:    stop = stopsec[LEFT_LWR_ARM]; break;
    case RIGHT_UPR_ARM:   stop = stopsec[RIGHT_LWR_ARM]; break;
    case LEFT_UPR_LEG:    stop = stopsec[LEFT_LWR_LEG]; break;
    case RIGHT_UPR_LEG:   stop = stopsec[RIGHT_LWR_LEG]; break;
}

xlate[0][0] = 1.0;
xlate[0][1] = 0.0;
xlate[0][2] = 0.0;
xlate[0][3] = -(Man[start][0][0] +
    (Man[start][1][0] - Man[start][0][0]) / 2.0);
xlate[1][0] = 0.0;
xlate[1][1] = 1.0;
xlate[1][2] = 0.0;
xlate[1][3] = -(Man[start][0][1] +
    (Man[start][3][1] - Man[start][0][1]) / 2.0);
xlate[2][0] = 0.0;
xlate[2][1] = 0.0;
xlate[2][2] = 1.0;
xlate[2][3] = -(Man[start][0][2] +
    (Man[start][3][2] - Man[start][0][2]) / 2.0);
xlate[3][0] = 0.0;
xlate[3][1] = 0.0;
xlate[3][2] = 0.0;
xlate[3][3] = 1.0;

rotat[0][0] = 1.0;
rotat[0][1] = 0.0;
rotat[0][2] = 0.0;
rotat[0][3] = 0.0;
rotat[1][0] = 0.0;
rotat[1][1] = cos(angle);
rotat[1][2] = -sin(angle);
rotat[1][3] = 0.0;
rotat[2][0] = 0.0;
rotat[2][1] = sin(angle);
rotat[2][2] = cos(angle);
rotat[2][3] = 0.0;
rotat[3][0] = 0.0;
rotat[3][1] = 0.0;
rotat[3][2] = 0.0;
rotat[3][3] = 1.0;

matrixMult(&rotat, &xlate, &tr);

xlate[0][0] = 1.0;
xlate[0][1] = 0.0;
xlate[0][2] = 0.0;
xlate[0][3] = -xlate[0][3];
xlate[1][0] = 0.0;
xlate[1][1] = 1.0;
xlate[1][2] = 0.0;
xlate[1][3] = -xlate[1][3];
xlate[2][0] = 0.0;
xlate[2][1] = 0.0;
xlate[2][2] = 1.0;
xlate[2][3] = -xlate[2][3];
xlate[3][0] = 0.0;
xlate[3][1] = 0.0;
xlate[3][2] = 0.0;
xlate[3][3] = 1.0;

matrixMult(&xlate, &tr, &trt);

```



```

matrixMult2(&strt, &vt, &vt2);
matrixMult2(&strt, &ht, &ht2);
matrixMult2(&strt, &At, &At2);
matrixMult2(&strt, &Bt, &Bt2);

/* Transform limb cross section points */
for (i=0; i<8; i++)
{
    matrixMult2(&strt, &Man[k2][i], &Man2[k2][i]);
    for (j=0; j<4; j++)
        Man[k2][i][j] = Man2[k2][i][j];
}

for(i=0;i<4;i++){
    v[i] = vt2[i];
    h[i] = ht2[i];
    A[i] = At2[i];
    B[i] = Bt2[i];
}

void matrixMult (float (*m1)[4][4], float (*m2)[4][4], float (*result)[4][4])
{
    int i,j,k;
    /* Multiply matrices m1 and m2 and put the the product in result. */
    /* m1 and m2 must be 4 by 4 matrices */
    for (i=0; i<4; i++)
    {
        for (j=0; j<4; j++)
        {
            (*result)[i][j] = 0;
            for (k=0; k<4; k++)
                (*result)[i][j] = (*result)[i][j] + (*m1)[i][k] * (*m2)[k][j];
        }
    }
}

void matrixMult2 (float (*m1)[4][4], float (*colvect)[4], float (*result)[4])
{
    int i,j;
    /* Multiply 4 by 4 matrix m1 by the column vector colvect and put the */
    /* product in result. */
    for (i=0; i<4; i++)
    {
        (*result)[i] = 0;
        for (j=0; j<4; j++)
            (*result)[i] = (*result)[i] + (*m1)[i][j] * (*colvect)[j];
    }
}

```

Virtual Reality and V&V Tools

Some of the tools in this category can be found in the draft report for V&V.

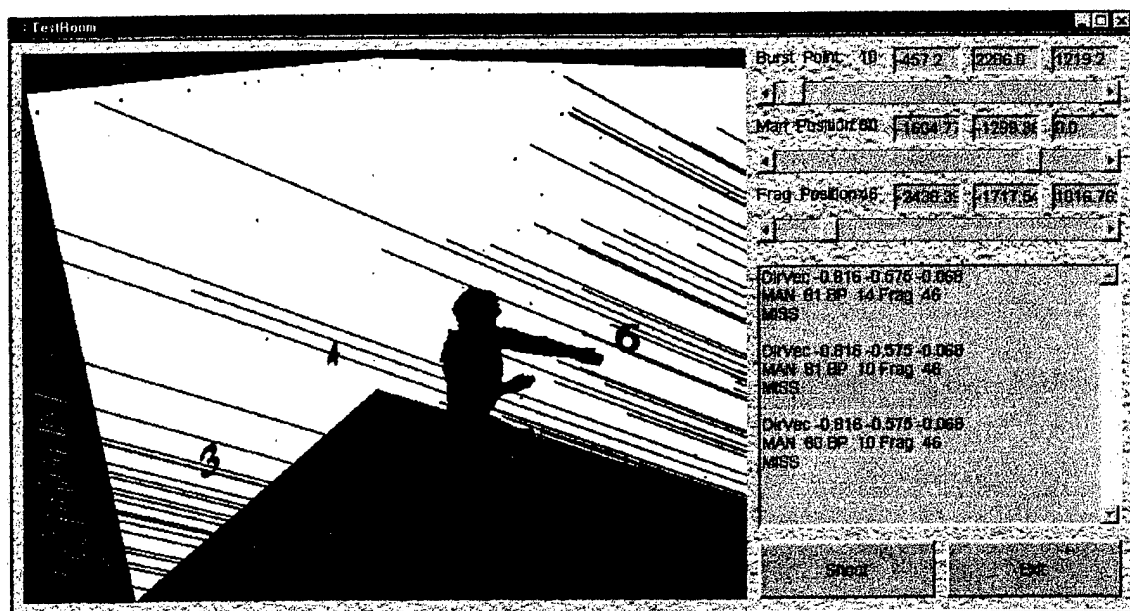


Figure 11- The Interactive Analysis environment in ProspectV2

However, the python script and an example image will be presented to show that a full analysis method exist on the PC without use of BRL-CAD raytracing. In fact, once the geometry is correct and used properly any raytrace that will give an accurate impact flag on the appropriate polygon can be used. However, for uparmoring work it would be prudent to use the same methodologies found in the Lethality / Vulnerability world to describe the geometry and the materials.

Figure 11 shows the user environment generated by the script in the next paragraph which is used to create the testroom simulation. This allows for full "flythrough" capability and the ability to shoot any frag from the selected burst point and man position.

```

from GuiTools import *
from Gui3DTools import *
from SEtreeTools import *
from PyOpenGL import *
from array import *
from math import *
import string

def Shutdown(WidgArg, UserArg): ProspectExit();

def Refresh(WinWidg, JAV):
    if SEtree.Reconcile() == 0:
        Gui3D.RenderSetup(WinWidg);
        Gui3D.Clear(WinWidg);

        curbp = GuiSlider.GetValue(SL1_Widg);
        curman = GuiSlider.GetValue(SL2_Widg);
        curfrag = GuiSlider.GetValue(SL3_Widg);

        curbpkey = num_bp - curbp;
        curmankey = num_man - curman;
        curfragkey = num_frag - curfrag;

        glColor4f(1.0,0.0,0.0,1.0);
        glLineWidth(2.0);
        keysarrayfrag = frag.keys()
        keysarraybp = bp.keys()
        keysarrayman = men.keys()
        i = len(keysarrayfrag)
        i = i - 1

        burstPoint = frag[keysarrayfrag[i]]
        newbp = bp[keysarraybp[curbpkey]]
        manpos = men[keysarrayman[curmankey]]

        node = SEtree.GetNodeByAbsName("TestRoom/man1");
        SEtree.SetPosition(node, (-manpos[0]/1000.0+0.2676144, manpos[2]/1000.0-0.405384, manpos[1]/1000.0-0.1231392));

        numFrag = burstPoint[0]
        numFrag = int(numFrag) - 1

        while numFrag >= 0:
            tmp1 = frag[keysarrayfrag[numFrag]]
            glBegin(GL_LINES);
            glVertex3f((-newbp[0]/1000.0), (newbp[2]/1000.0), (newbp[1]/1000.0));
            glVertex3f((tmp1[0]*-0.0254), (tmp1[2]*0.0254), (tmp1[1]*0.0254));
            glEnd();
            numFrag = numFrag - 1
            tmp1 = ();

            tmp1 = frag[keysarrayfrag[curfragkey]]
            glBegin(GL_LINES);
            glColor3f(0.0,1.0,0.0);
            glVertex3f((-newbp[0]/1000.0), (newbp[2]/1000.0), (newbp[1]/1000.0));
            glVertex3f((tmp1[0]*-0.0254), (tmp1[2]*0.0254), (tmp1[1]*0.0254));
            glEnd();

            Gui3D.RenderSEtree(WinWidg, NULL);
            Gui3D.SwapBuffers(WinWidg);

        return 0;

def Redraw(WidgArg, UserArg):
    Gui3D.SetCallbacks(WidgArg, NO_CALLBACK, Gui3D.FlyerMB, Gui3D.FlyerMM, NO_USERARG);
    Gui3D.SetProjection(WidgArg, 60.0, 45.0, 0.1, 1000.0);
    Gui3D.SetViewpoint(WidgArg, (-11.0, 5.0, 0.0), (0.0, -30.0, -90.0));
    Gui3D.SetLight(WidgArg, 0, (1.0, 0.9, 0.8, 0.0), (1.0, 1.0, 1.0));
    GuiProc.AddTimer(0.01, Refresh, (WidgArg, JAV));
    return;

```

```

def JAVControl(Node, FrameNum, UserArg):
    Pos = SEtree.GetPosition(Node)
    Vel = SEtree.GetVelocity(Node)
    NewVel = Vel[0];
    if Pos[0] > 0.0:
        NewVel = 0.00;
        SEtree.SetAttitude(Node, (0.0, 90.0, 0.0));
    elif Pos[0] < 0.0:
        NewVel = 1.10;
        SEtree.SetAttitude(Node, (0.0, 0.0, 90.0));
    SEtree.SetVelocity(Node, (NewVel, Vel[1], Vel[2]));
    return;

def BP_CB(WidgArg, UserArg):
    curbp = GuiSlider.GetValue(SL1_Widg);
    curbp = curbp + 1;
    string = str(curbp);
    GuiLabel.Retitle(BPNumLabel, string);

    curbpkey = num_bp - (curbp-1)
    keysarraybp = bp.keys()
    newbp = bp[keysarraybp[curbpkey]];

    GuiScalarFloat.SetValue(BPwidgX, newbp[0]);
    GuiScalarFloat.SetValue(BPwidgY, newbp[1]);
    GuiScalarFloat.SetValue(BPwidgZ, newbp[2]);

    Refresh((GLWidg, JAV));
    return;

def Man_CB(WidgArg, UserArg):
    curman = GuiSlider.GetValue(SL2_Widg);
    curman = curman + 1;
    string = str(curman);

    GuiLabel.Retitle(ManNumLabel, string);

    curmankey = num_men - (curman-1)
    keysarraymen = men.keys()
    newman = men[keysarraymen[curmankey]];

    GuiScalarFloat.SetValue(ManwidgX, newman[0]);
    GuiScalarFloat.SetValue(ManwidgY, newman[1]);
    GuiScalarFloat.SetValue(ManwidgZ, newman[2]);

    Refresh((GLWidg, JAV));
    return;

def Frag_CB(WidgArg, UserArg):
    curfrag = GuiSlider.GetValue(SL3_Widg);
    curfrag = curfrag + 1;
    string = str(curfrag);

    GuiLabel.Retitle(FragNumLabel, string);

    curfragkey = num_frag - (curfrag-1)
    keysarrayfrag = frag.keys()
    newfrag = frag[keysarrayfrag[curfragkey]];

    GuiScalarFloat.SetValue(FragwidgX, newfrag[0]*25.4);
    GuiScalarFloat.SetValue(FragwidgY, newfrag[1]*25.4);
    GuiScalarFloat.SetValue(FragwidgZ, newfrag[2]*25.4);

    Refresh((GLWidg, JAV));
    return;

def Shoot_CB(WidgArg, UserArg):
    Inpoint2 = {}
    Outpoint2 = {}
    DirVec = {}
    NewDirVec = {}

    curbp = GuiSlider.GetValue(SL1_Widg);
    curman = GuiSlider.GetValue(SL2_Widg);
    curfrag = GuiSlider.GetValue(SL3_Widg);

    curbpkey = num_bp - curbp;
    curmankey = num_men - curman;
    curfragkey = num_frag - curfrag;

    keysarrayfrag = frag.keys()
    keysarraybp = bp.keys()
    keysarraymen = men.keys()

    i = len(keysarrayfrag)
    i = i - 1

    burstPoint = frag[keysarrayfrag[i]]
    newbp = bp[keysarraybp[curbpkey]]
    newfrag = frag[keysarrayfrag[curfragkey]]
    manpos = men[keysarraymen[curmankey]]

    # DirVec[0] = newfrag[0]*-0.0254 + newbp[0]/1000.0
    # DirVec[1] = newfrag[1]*0.0254 - newbp[1]/1000.0
    # DirVec[2] = newfrag[2]*0.0254 - newbp[2]/1000.0

```

```

DirVec[0] = newfrag[0]*-0.0254 + 0.0/1000.0
DirVec[1] = newfrag[1]*0.0254 - 0.0/1000.0
DirVec[2] = newfrag[2]*0.0254 - 1219.2/1000.0

magsqr = DirVec[0]*DirVec[0]+ DirVec[1]*DirVec[1]+DirVec[2]*DirVec[2]
magn = sqrt(magsqr)

NewDirVec[0] = DirVec[0] / magn;
NewDirVec[1] = DirVec[1] / magn;
NewDirVec[2] = DirVec[2] / magn;

string = "DirVec %6.3f %6.3f %6.3f" % (-NewDirVec[0],NewDirVec[1],NewDirVec[2])
GuiScrolledList.AppendValue(ScrolledList_Widg,string);
string = ();

Ray = ((-newbp[0]/1000.0,newbp[2]/1000.0,newbp[1]/1000.0),(NewDirVec[0],NewDirVec[2],NewDirVec[1]));
RTNode = SETree.GetNodeByAbsName("TestRoom/man1");
Hit,Node,Inpoint,Inorm = SETree.RaytraceSubtree(RTNode,Ray);
if Hit == 1:
    Ray = ((Inpoint[0]+NewDirVec[0]*100.0,Inpoint[1]+NewDirVec[2]*100.0,Inpoint[2]+NewDirVec[1]*100.0),(-
NewDirVec[0],-NewDirVec[2],-NewDirVec[1]));
    Hit2,Node2,Outpoint,Inorm2 = SETree.RaytraceSubtree(RTNode,Ray);
    string = "MAN %3d BP %3d Frag %3d" % (curman+1,curbp+1,curfrag+1)
    GuiScrolledList.AppendValue(ScrolledList_Widg,string);
    string = ();
    GuiScrolledList.AppendValue(ScrolledList_Widg,"HIT");
    Inpoint2[0] = -Inpoint[0]*1000.0 + 267.6144 - manpos[0]
    Inpoint2[1] = Inpoint[1]*1000.0 + 405.3840 - manpos[2]
    Inpoint2[2] = Inpoint[2]*1000.0 + 123.1392 - manpos[1]
    string = "Inhit %6.3f %6.3f %6.3f" % (Inpoint2[0],Inpoint2[2],Inpoint2[1])
    GuiScrolledList.AppendValue(ScrolledList_Widg,string);
    string = ();
    if Hit2 == 1:
        Outpoint2[0] = -Outpoint[0]*1000.0 + 267.6144 - manpos[0]
        Outpoint2[1] = Outpoint[1]*1000.0 + 405.3840 - manpos[2]
        Outpoint2[2] = Outpoint[2]*1000.0 + 123.1392 - manpos[1]
        string = "Outhit %6.3f %6.3f %6.3f" % (Outpoint2[0],Outpoint2[2],Outpoint2[1])
        GuiScrolledList.AppendValue(ScrolledList_Widg,string);
        string = ();
    elif Hit2 == 0:
        GuiScrolledList.AppendValue(ScrolledList_Widg,"NO OUTPOINT?!!");
elif Hit == 0:
    string = "MAN %3d BP %3d Frag %3d" % (curman+1,curbp+1,curfrag+1)
    GuiScrolledList.AppendValue(ScrolledList_Widg,string);
    string = ();
    GuiScrolledList.AppendValue(ScrolledList_Widg,"MISS");

GuiScrolledList.AppendValue(ScrolledList_Widg,"");
return;

def OpenDataFile(associativeArray, fileName):
    k = 0
    for line in open(fileName, 'r').readlines():
        tmp = string.split(line)
        for j in range(len(tmp)):
            tmp[j] = float(tmp[j])
        associativeArray[k] = tmp
        k = k + 1
    return k;

def main(ArgC, ArgV):
    global JAV;
    global frag, bp,men, num_fraags, num_bp, num_men;
    global SL1_Widg, BPNumLabel,GLWidg;
    global SL2_Widg, ManNumLabel;
    global SL3_Widg, FragNumLabel;
    global BPwidgX, BPwidgY, BPwidgZ;
    global ManwidgX, ManwidgY, ManwidgZ;
    global FragwidgX, FragwidgY, FragwidgZ;
    global ScrolledList_Widg

    frag = {}
    bp = {}
    men = {}

    num_fraags = OpenDataFile(frag, "6dc05.3dp")
    # num_fraags = OpenDataFile(frag, "chris.3dp")

    num_bp = OpenDataFile(bp, "bpos.txt")
    num_men = OpenDataFile(men,"manpos.txt")

    num_fraags = num_fraags - 2
    num_bp = num_bp - 1
    num_men = num_men - 1

    SETree.AddSubtree(NULL, "TestRoom", "TestRoom/SETreeBuild");
    JAV = SETree.GetNodeByAbsName("TestRoom/missile");
    #SETree.SetVelocity(JAV, (1.0, 0.0, 0.0));
    SETree.SetAttitude(JAV, (0.0, 0.0, 90.0));

    WinWidg = GuiWindow.Create("TestRoom", 990, 500, "lighttest.gif", Shutdown, NO_USERARG);

    GLWidg = Gui3D.Create(WinWidg,10,10,640,480,Redraw,NO_CALLBACK,NO_CALLBACK,JAV );

```

```

GuiButton.Create(WinWidg,660,440,155,50,"Shoot",Shoot_CB,NO_USERARG);
GuiButton.Create(WinWidg,825,440,155,50,"Exit",Shutdown,NO_USERARG);

GuiLabel.Create(WinWidg,660,10,"Burst Point:");
BPNumLabel = GuiLabel.Create(WinWidg,750,10,"1");
SL1_Widg = GuiSlider.Create(WinWidg,660,40,320,20,HORIZONTAL,0,num_bp,BP_CB,NO_USERARG);

GuiLabel.Create(WinWidg,660,70,"Man Position:");
ManNumLabel = GuiLabel.Create(WinWidg,750,70,"1");
SL2_Widg = GuiSlider.Create(WinWidg,660,100,320,20,HORIZONTAL,0,num_men,Man_CB,NO_USERARG);

GuiLabel.Create(WinWidg,660,130,"Frag Position:");
FragNumLabel = GuiLabel.Create(WinWidg,750,130,"1");
SL3_Widg = GuiSlider.Create(WinWidg,660,160,320,20,HORIZONTAL,0,num_frgs,Frag_CB,NO_USERARG);

BPwidgX = GuiScalarFloat.Create(WinWidg,780,10,60,NO_EDIT,NO_CALLBACK,NO_USERARG);
BPwidgY = GuiScalarFloat.Create(WinWidg,850,10,60,NO_EDIT,NO_CALLBACK,NO_USERARG);
BPwidgZ = GuiScalarFloat.Create(WinWidg,920,10,60,NO_EDIT,NO_CALLBACK,NO_USERARG);

ManwidgX = GuiScalarFloat.Create(WinWidg,780,70,60,NO_EDIT,NO_CALLBACK,NO_USERARG);
ManwidgY = GuiScalarFloat.Create(WinWidg,850,70,60,NO_EDIT,NO_CALLBACK,NO_USERARG);
ManwidgZ = GuiScalarFloat.Create(WinWidg,920,70,60,NO_EDIT,NO_CALLBACK,NO_USERARG);

FragwidgX = GuiScalarFloat.Create(WinWidg,780,130,60,NO_EDIT,NO_CALLBACK,NO_USERARG);
FragwidgY = GuiScalarFloat.Create(WinWidg,850,130,60,NO_EDIT,NO_CALLBACK,NO_USERARG);
FragwidgZ = GuiScalarFloat.Create(WinWidg,920,130,60,NO_EDIT,NO_CALLBACK,NO_USERARG);

ScrolledList_Widg = GuiScrolledList.Create(WinWidg,660,200,320,230,NO_CALLBACK,NO_CALLBACK,NO_USERARG);

GuiSlider.SetValue(SL1_Widg,0);
GuiSlider.SetValue(SL2_Widg,0);
GuiSlider.SetValue(SL3_Widg,0);

GuiTree.Map(WinWidg);

```

Conclusions

This report documents the efforts under this task. All data is in the possession of the COTR for the respective tasks. This work has produced several tools which should help add capability in the assessment of Incapacitation in MOUT analysis. Many of the modules created under this effort have applicability to other Army work outside of this program. It is hoped that these tools will be widely distributed to any user who has a valid requirement.

APPENDIX I - V&V Plan

VERIFICATION AND VALIDATION PLAN FOR 3DPIMMSMAN COMPUTER CODE

1.0 Purpose

In FY99, the US Army Aviation and Missile Command (AMCOM) has been tasked by the MPIM/SRAW Program to provide the Verification and Validation (V&V) of the 3DPimmsMan simulation. AMSAA agreed to critique the AMCOM efforts since the effort benefits not only the Army MPIM/SRAW program but also any future systems that will need to evaluate their performance against bunkers and buildings. This document will serve as the baseline V&V Plan.

2.0 Background

2.1 General Information

The utilization of simulations provides the government a valuable tool to perform system's analyses to characterize the performance of a missile system. Given the Department of Defense (DoD) shrinking budgets simulations have become the mainstay to study and evaluate system performance. The MPIM/SRAW program is a shoulder launched weapon system designed to incapacitate personnel in a bunker, concrete or triple brick building or light armored vehicle (MOUT scenario). Since, MOUT operations have become a larger percentage of the projected types of combat situations which may be presented to the U.S. military, high resolution tools are required to evaluate missile system performance. An illustration of these types of targets can be seen in Figure 1-3.

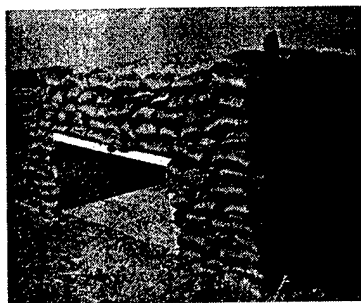


Figure 1 – A Bunker

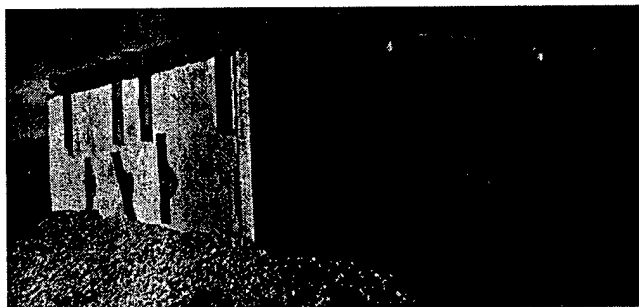


Figure – 2 A Room Target with Door

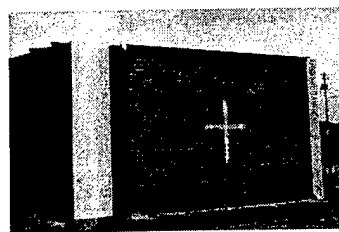


Figure- 3 Triple brick

In the past the Probability of Incapacitation Methodology for Masonry Structures (PIMMS) has been the code of choice by AMSAA for the lethality analysis of bunkers and buildings. Fragmentation data is collected at the time of a test by lining the bunker or room with witness panels which record the location of each lethal fragment. Utilizing the PIMMS code any fragment that completely penetrates the witness panels is counted as lethal. PIMMS

models the crouching man as a cylinder and uses a fragment density number per steradian in order to calculate Probability of Incapacitation (Pi) for each possible man location within the room or bunker. The code then marches the burst point location around the room in one foot increments and calculates the Pi for each possible burst point location. These 256 burst points are averaged to come up with an overall Pi for each set of test data. This code is a legacy code but there has been V&V of a Point and Click and a Light Table data interface program to the 2D code (See attached Appendix A, report by Windsor Jones, AMSAA)) The original methodology and requirements for the PIMMS code are spelled out in CSC TR-81-5692. (Probability of Incapacitation Methodology for Masonry structures)

In FY97 AMCOM developed an improved version of the PIMMS code. This code rejects the assumption of a lethal fragment which is present in the 2D PIMMS methodology. Instead an automated wound ballistics approach is implemented to decide if a fragment trace would produce incapacitation of a human in the path. The 3DpimmsMan employs the BRL Raytrace software and the BRL ComputerMan code in a three Dimensional environment. The Documentation for this software can be found in Appendix I, UAH Report #5 -34381. The data collection procedure is done exactly the same as it was for the 2-D environment. However, since the lethal fragment assumption is being rejected, any roughly equivalent thickness panels could be used if methods are utilized to determine fragment velocity distribution. The BRL Raytrace software provides the capability to trace a fragment from a given burst point to it's exit x,y and z location within the room. The BRL ComputerMan code consists of a tissue data base and velocity retardation coefficients for a standard man. This code in it's standard GUI implementation allows you to input entry and exit wound locations for a given fragment on a single man and output a Pi. The 3DpimmsMan code is essentially a simulation which provides the bookkeeping required to take the x,y, and z locations of each fragment within the room, raytrace each and every fragment from the estimated burst point location, track the fragment through an actual 3-D tissue database of a man (derived from a batch mode implementation of the ComputerMan code). The pmssub subroutine keeps track of the fragment velocity retardation as the fragment travels through different tissue types and the output reports the extent of the man's injuries and incapacitation levels. This is done for each fragment, for each possible burst point location and for all possible man positions within the room. The same procedure is followed for averaging the Pi's for each possible burst point location.

2.2 Configuration Control

During the V&V process a Configuration Control Board will be implemented. Ms. Edith Crow of AMCOM will serve as the Chairperson with Mr. Brian Sabourin serving as Project Office lead. Mr. Glenn Romanczuk and Mr. Chris Pitts will also be on the board. The Board will determine what type of revision control is used and the available version of the code.

The 3Dpimms methodology does draw from ARL generated and controlled software tools. These two codes are the BRLCAD set of Raytrace tools and the wound ballistics model BRL ComputerMan. These codes are controlled and distributed by ARL under an ARL Licence agreement. The Versions of the libraries for these two codes will be spelled out in all documentation and will be required for the use of this tool.

2.2 Identification of Agencies

Systems Simulation and Development Directorate, Aeroballistics Analysis Functional Area, (Ms. Edith W. Crow and UAH support Contractors) were tasked to support the MPIM Project to provide lethality analysis. As a result of providing support in this area over the years question were raised by customers about what other analysis tools were available which would provide the most realistic results for fragmentation and allow for additional insults to be credited. The evolution of the analysis brought about the investigation of employing the BRL-CAD Raytrace software and the ARL ComputerMan model since this would provide the first opportunity to do this analysis in a 3-Dimensional environment. This modification allows floor fragments to enter the analytical process as well as allow other secondary incapacitation insults to be evaluated. The following table shows the responsibilities of each agency involved.

AGENCY	Software	RESPONSIBILITIES
AMCOM	3DpimmsMan	Verification and Validation
EAC		Accreditation
AMSAA	2 D PIMMS	Verification and Validation
EAC		Accreditation
ARL	BRL Raytrace ComputerMan	Accepted Methodologies

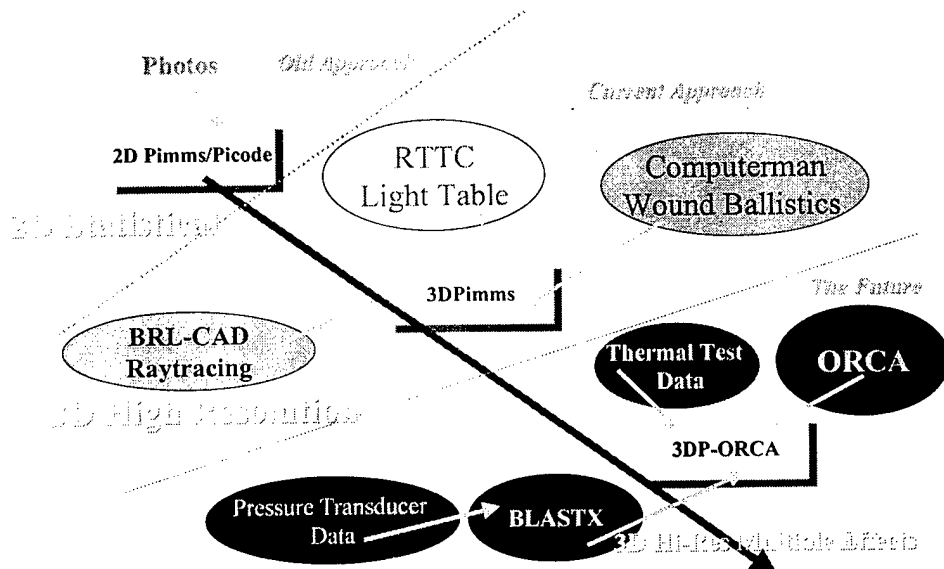


Figure 4 shows the proposed progression of the Analysis methodology.

3.0 Intended Use of 3DpimmsMan

The major program objectives are to provide a state-of-the-art lethality analysis toolkit for the evaluation of personnel incapacitation in bunkers and buildings. This tool will provide the many-on-many level of analysis which is required. This new analysis tool will pave the way for the evaluation of other insults such as thermal, overpressure, toxic gases, etc. The new tool also more accurately describes the test data that has been collected and applies it to an actual 3-D room. The 3D methodology allows for the evaluation of arena test data (JMEM) which could not be evaluated using the 2-D PIMMS code. Figure 4 shows the evolution of the analysis code from a 2 dimensional statistical tool into a high resolution, physics based incapacitation code for multiple insults. It is important to note that if the ORCA code which is also undergoing VV&A in 1999, is not ready for inclusion into this analytical process then the underlying submodels which ORCA draws upon could be used to extend the 3Dpimms code to handle the additional insults required.

4.0 Data Collection

Data collection methodology for the 3DpimmsMan is currently done in accordance with the 2D PIMMS data collection. PIMMS data is collected and analyzed in a specific manner, reference 1, and not in the more familiar manner used in the Joint Munitions Effectiveness manual (JMEM). The fragmentation data used for PIMMS incorporates wall debris and lethal fragments. The fragmentation data does not address fragment mass, velocity or shape. However, the Light table collection technique does provide the analyst with the presented area of each fragment. Only the number of lethal fragments in a geometrical region are used in the 2D code. However, the 3D software would be capable of using other types of data such as the arena test data (JMEM Data). Fragment mass, velocity and shape data could be used to characterize each set of test data if arena test data is available. Also, the analyst is called upon to enter an average mass and velocity of the fragments which can be gathered from hydrocode calculation or from experimental techniques.

5.0 Verification and Validation (V&V) Plan

According to Department of the Army Pamphlet 5-11, verification is the process of determining that a model or simulation represents the developer's conceptual description and specifications and meets the needs stated in the requirements document. The verification process thereby establishes whether the model or simulation code and logic correctly perform the intended functions. Validation is the process of determining the extent to which a model or simulation accurately represents the real world phenomena from the perspective of the intended use of the model or simulation. In this section, an attempt will be made to formulate a plan of action that will be taken to effectively verify and validate 3DpimmsMan for its use as an evaluation tool for incapacitation of personnel in MOUT structures.

5.1 Definitions

In an effort to assist the reader in understanding the V&V process, an explanation of the techniques follows:

Documentation reviews - This review ensures that the design and specifications encompass the model or simulation requirements and that they represent a balanced and correct approach. It also includes reviewing the specification document, design documentation, and code to ensure that all of the requirements are addressed in an appropriate and complete manner.

Functional decomposition - Decomposing the model or simulation into functional components is often a great aid in the validation process. A detailed examination of the documentation, code and output to determine that validity of the decomposed model or simulation is executed. Then an analysis of how well the pieces fit together is accomplished, with the result being an overall validation of the model or simulation. Decomposition of the model or simulation should be sensitive to the intended use of the model or simulation as this may drive the functional split and the level to which the decomposition is done.

Algorithm check - This involves inspection of design documents to compare equation and algorithm methodology to outside documentation. A key issue here is determining whether the documented equations match those found in other publications or other successful model or simulations.

Sensitivity analysis - This is a check of the algorithms and code to ensure that the model or simulation is reacting to varying sets of input in an expected, mathematically predictable manner. These analyses include preparing and running tests to compare results for systematically varied sets of input data to see if the expected trends in output are demonstrated.

Units check - This is a check to ensure that the proper units of measure result from equations in the algorithms and code.

Graphics playback - This technique allows the analyst to see the model simulations behavior graphically. This is particularly useful for visualizing input fragment panel data, entry and exit wounds, and fragment distribution patterns.

This list of V&V techniques is not conclusive but represent a sufficient subset to be used in the analysis of 3DpimmsMan for accreditation.

5.2 3DpimmsMan

5.2.1 Description

Program approach and methodology summary

The methodology utilized can be summarized in three parts. The first part is the burst point logic and bookkeeping from the legacy code PIMMS. This part details the creation of man locations in the room and bunker. Also, it models the creation of burst points given the user inputs.

The second part of the methodology takes the input of the fragment location per panel and produces an x,y,z base upon the room coordinate frame. Then fragments are raytraced utilizing BRL-CAD Raytrace tools to produce an entry and exit wound path through a crouching man geometry in one of the possible man locations. Rst point versus the location of the man in the room. It is assumed that there is only one man at a random location in the room. Therefore it is not necessary to raytrace a full room. In fact, the code utilizes the man in a stationary manner and offsets the origin of the ray based upon the ray burst point versus the location of the man in the room. This is done for all fragments to determine which fragments strike person in that position given the fragment data. The crouching man geometry must be an exact match of the geometry utilized in the wound ballistics model ComputerMan. Figure 5 shows a rendering of the man geometry which will be verified.

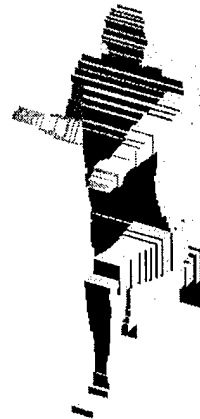


Figure 5 – The Bbman geometry

The third part is the incapacitation subroutine. This subroutine is a batch mode version of the ComputerMan routine. However, it has been augmented to handle several wound paths before the determination of incapacitation is output. The inputs that are required include the array of entry and exit points and the number of wounds on each man. There a a number of items which need to be verified to assure that the wound ballistics model is being properly employed. In the development of the subroutine version of pmssub a standalone version was also coded. This version will be run to check output of the pmssub routine.

5.2.2 Plan

Overall Verification and Validation Objectives

The overall V&V objectives are presented in the following paragraphs.

‘The overall objective of V&V of the 3DPimms is to ensure that the simulation models real-world behaviors and assesses the Probability of Incapacitation for personnel in masonry structures and Earth & Timber Bunkers to an acceptable level of accuracy in the many-on-many case. Meeting this requirement will provide an analytical tool which can be used for determination of a systems ability to meet a specified ROC requirements for Incapacitation probability.’

‘The process to achieve this overall objective will include exercising the simulations and comparing the results with the current 2D Pimms simulation given the limitations and assumptions inherent in the 2D methodology’. The primary comparison to the 2D methodology will explore those areas of similarity and will report probable explanations for differences which may be found.

Several elements are listed in the V&V Plan as items to be required for V&V. These items were simulation configuration management, documentation of the simulation, verification documentation and validation documentation.

5.2.3 V&V Test Matrix

In an effort to better assist the reader in understanding the process that will be used to V&V 3DpimmsMan, Figure 6 graphically describes the process. AMCOM will perform the V&V activities for the 3DpimmsMan and AMSAA’s 2D PIMMS V&V documentation will be used to support these results.

Under the modeling and simulation (M&S) acceptability criteria, nine requirement areas were identified which are required to V&V the 3Dpimms code.

Requirement Area	Description
1	Simulation must accurately load fragment witness panel data and convert the panel data to a room coordinate frame. Graphical 3 D representation must be able to represent these results.
2	Simulation must accurately position the man within the room

Requirement Area	Description
	and calculate man locations for different size rooms. User must prove that man location relative to burst point location is represented accurately and must also provide graphical output to represent these results.
3	Simulation must prove that fragment path , entry and exit wound position in the Crouching man frame of reference, data is accounted for properly in the call to the incapacitation subroutine. User must demonstrate this by a graphical representation of Raytrace position to support this function.
4	The simulation must accurately provide bookkeeping for Pi for each person within the room and demonstrate that arrays of data are loaded properly. Commented code loops and output should be used to explore this requirement.
5	The simulation must provide credible results that intuitively match expected results for a sample set of shots. This includes low and high fragment cases.
6	The simulation must accurately position the burst point matrix based upon the user inputs for size of room and step size.
7	Deviations from 2D results must be explained for a sample set of 3D data.
8	Simulation must accurately calculate the room coordinates for fragments from the input panel x,y data per panel.
9	Simulation must prove that the fragment input data to the incapacitation module is consistent with fragments witnessed in test.

Figure 6 - The V&V Requirements Table

5.3 V&V Process Summary

Figure 7 documents the specific parts of the 3dp.c code which must be evaluated using the methods in the V&V methods section. The description of each part of the code is presented and will be included in the documentation of the V & V efforts. The methods listed and maybe also a few additional techniques will be utilized to verify and validate the program modules which are listed in Figure 7. Graphical tools will be used when appropriate. Also, MathCad worksheets and code traces with print statements and loop counters will be used to test structure and verify functionality.

Algorithm/Module	Description	V&V Methods
Men_t Structure	Main structure which stores man locations and incapacitation	<ul style="list-style-type: none"> • Document Review • Funcional Decomposition • Algorithm Checks • Sensitivity Analysis • Units Checks • Graphics Playback
Read_config	Subroutine to read setup information	
Incap_calc	The main calculation subroutine given a config file and a frag file	
Load_frag_data	Reads the panel data and converts 2D to 3D to feed raytrace	
Calc_man_positions	Calculates the positions of the men in the room based on room size.	
Calc_bp_positions	Calculates the burst point locations within the room based on step sizes and wall sizes.	
Rt_shootray	Given inputs shoots ray	
Cman_incap	Calculates the incapacitation given a number of hits and the entry and exit points.	
hit	Rt_shootray subroutine for inpacts on geometry	
miss	Rt_shootray subroutine for ray misses on geometry	

Print_front_matrix Print_side_matrix Print_front_twoman_matrix Print_sniper_matrix Print_side_twoman Print_side_sniper	Prints the given output matrix	
uniform	Returns a uniform distribution between the min and max values that are input	

Figure 7 - The Specific routines in 3DP to be evaluated for verification and validation

Figure 8 is a logic diagram of the entire 3DP.c code. This is a statement by statement description of the execution logic in 3DP. This logic will be checked with code traces using counters and output of matrix and structure information. This data will be visualized with a number of techniques.


```

1) read config file
  a) reads in config file info: min frag mass, max frag mass, min frag velocity, max frag velocity, frag shape factor,
    frag density, man geometry flag (crouching or standing), man protection value, room flag and wall sizes
    (front and side), room step size, impact flag, twoman flag, sniper flag, use floor flag, attack angle
  b) set up frag array with uniform distribution of mass and velocity
2) load appropriate mged file and prep it for ray tracing
  a) load either crouching or standing mged man model
3) load frag file
  a) reads in frag data from various formats (old 2d, 2d gin, 3dp)
4) set up run monitor file and info
5) set up ray trace parameters
6) calculate positions of men in room based on room size and man size
  a) based on man width and man depth and front wall length and side wall length
  b) calculate positions for each man inside room with equal spacing from walls
7) do front shot
  a) calculate burst point locations in room based on step size
    I. based on step size and room size
    II. calculate positions for each burst point
    III. set flag for feasibility (checked with attack angle)
  b) set nbp (number of feasible burst points for averaging)
  c) front shot loop
    for each man position in the room {
      for each burst point location in the room {
        if it is a feasible burst point {
          calculate burst point origin relative to man position and translate to match mged man
          I. using man offset values from the mged file calculate burst point origin as
             MGED MAN OFFSET - (ROOM MAN LOCATION - BURST POINTLOCATION)
          set number of hits to 0 for this man at this bp
          for each frag {
            calculate directional cosines for frag path from burst point origin
            I. directional cosines calculated from initial burst point location and
               impact location of frag on witness panels (rotated 90 degrees for side shot)
            fire shotline
            if shotline hit man {
              save X,Y,Z for inhit and outhit locations and increment hit counter
            }
          }
          assign incapacitation value for this man and this burst point location with computer man subroutine
          and inhit/outhit locations, calculate total incap value for this man
        }
      }
    }
    divide total incap value for this man by nbp
    update run monitor file
  d) calculate total incapacitation value for front shot
8) do side shot (just like front shot only rotated 90 degrees)
9) print output
10) remove run monitor file
11) exit program

```

Figure 8 – The logic diagram for the 3DP.c code

5.4 2D PIMMS

5.4.1 History

The PICODE and PINFIB programs were designed and developed by Mr. Larry Losie of BRL. The PICODE program provides incapacitation probabilities for attacks on personnel inside masonry structures using impact fuzed munitions. An impact fuzed munition bursts at the point of contact with the exterior wall of the structure. The PINFIB program evaluates a program identical to that of PICODE except delay fuzed munitions are used for the attack. Computer Sciences Corporation of Huntsville, Alabama was placed under contract to produce the PIMMS program by merging or combining PICODE and PINFIB, and to provide documentation for the new program, PIMMS.

The PINFIB program was used as the basic structure for the PIMMS program. Logic and algorithms peculiar to PICODE were placed in the existing PINFIB code. These logic and algorithm were fully integrated into the PINFIB program logic to avoid producing a new program consisting of a collection of "patches".

As a result, the PIMMS program is the integration rather than the attachment of PICODE and PINFIB. The PIMMS program retains both the capability and flexibility of PICODE and PINFIB. Its code is well commented and it has convenient input data structure. Only minor improvements were made to the program logic provided to CSC. These changes were primarily cosmetic in nature.

5.4.2 Methodology

The room is filled with men for each attack. A cylinder of specified diameter and vertical cross-sectional area is used to represent a man. A sequence of burst points at a fixed interval along the wall is evaluated in a wall attack using impact fuzed munitions. A matrix of burst points covering the room (in the horizontal plane) is evaluated when delay fuzed munitions are used in a wall attack. Attack orientation or direction relative to the wall is an input value. Fragment data about the burst point are input in five-degree zones relative to the attack or shot direction. For each munition burst point, an incapacitation determination is made for each man in the room. A man is considered incapacitated when struck by one or more lethal fragments. When this criterion is not satisfied, the man is considered undamaged by the burst. Both the number of incapacitated men and their room locations are stored for each munition burst point. Attacks on each wall of the room are evaluated.

After the attack has been evaluated, probability of incapacitation data are generated for each room wall. Probability of incapacitation for a randomly located man from a given burst point is computed as the ratio of the number of incapacitated men to the number of

men in the room. Average and cumulative average probability of incapacitation data for a randomly located man are also provided for each wall attack.

Two of the man locations in the room are designated as the two man firing position. Three of the man locations are designated as possible locations for a sniper. Probability of incapacitation data are generated for the sniper and the two man firing team as a user option. The nominal configuration of the structure is an enclosed room (without windows or doors). However, when impact fuzed munitions are used in the attack and the sniper or two man options are selected, the sniper or two man incapacitation probabilities are computed both with and without a window in the front wall.

5.4.3 PIMMS V&V

The PIMMS VV&A is concurrently being completed by EAC. The original V&V plan and V&V documentation was written by Mr. Windsor Jones of AMSAA and was near completion when the effort was shelved due to program reorganization. Ms. Kathy Fontaine, EAC, has resurrected this effort and it is near completion at this time.

6.0 Accreditation Plan

EAC will be responsible for the accreditation plan and actual accreditation. As the system evaluator EAC is responsible to accredit each simulation for fitness to this task.

6.1 Acceptability Criteria

This section will address the criteria that 3DpimmsMan must meet to determine if it is suitable for its intended use. According to DA PAM 5-11, "Verification, Validation and Accreditation of Army model and simulation", failure of a model or simulation in achieving a particular acceptability criterion does not automatically result in the model or simulation not being accredited. Such an occurrence may result in an evaluation of the criticality of the criterion to overall success and merely serve to restrict the range of applicability of the problem. The following criteria will be used to determine 3DpimmsMan acceptability in the analytical community as an evaluation tool:

- 3DpimmsMan is suitable for determining the effectiveness of indirect fire munitions against personnel within masonry structures, i.e., reinforced concrete, brick and earth and timber bunkers.
- The definition of incapacitation used in 3DpimmsMan is sufficient in accessing the degradation level of personnel located within masonry structures and is a higher resolution calculation than that achieved in 2DPimms.

DRAFT

- The output of 3DpimmsMan, quantitatively and graphical, may be used clearly, adequately and appropriately to address how well an indirect fire munition performs against personnel within masonry structures.
- Required data values are well defined and data sources for obtaining data have been identified.
- The algorithms, methodology and environment representations are functionally adequate to address the issues.

EAC will be the agency responsible for addressing the stated criteria and performing the accreditation of 3DpimmsMan for its intended use. The efforts of accrediting 3DpimmsMan will be documented in a VV&A report.

7.0 Milestone schedule

The following is the proposed schedule that will be followed for the verification, validation and accreditation of 3DpimmsMan:

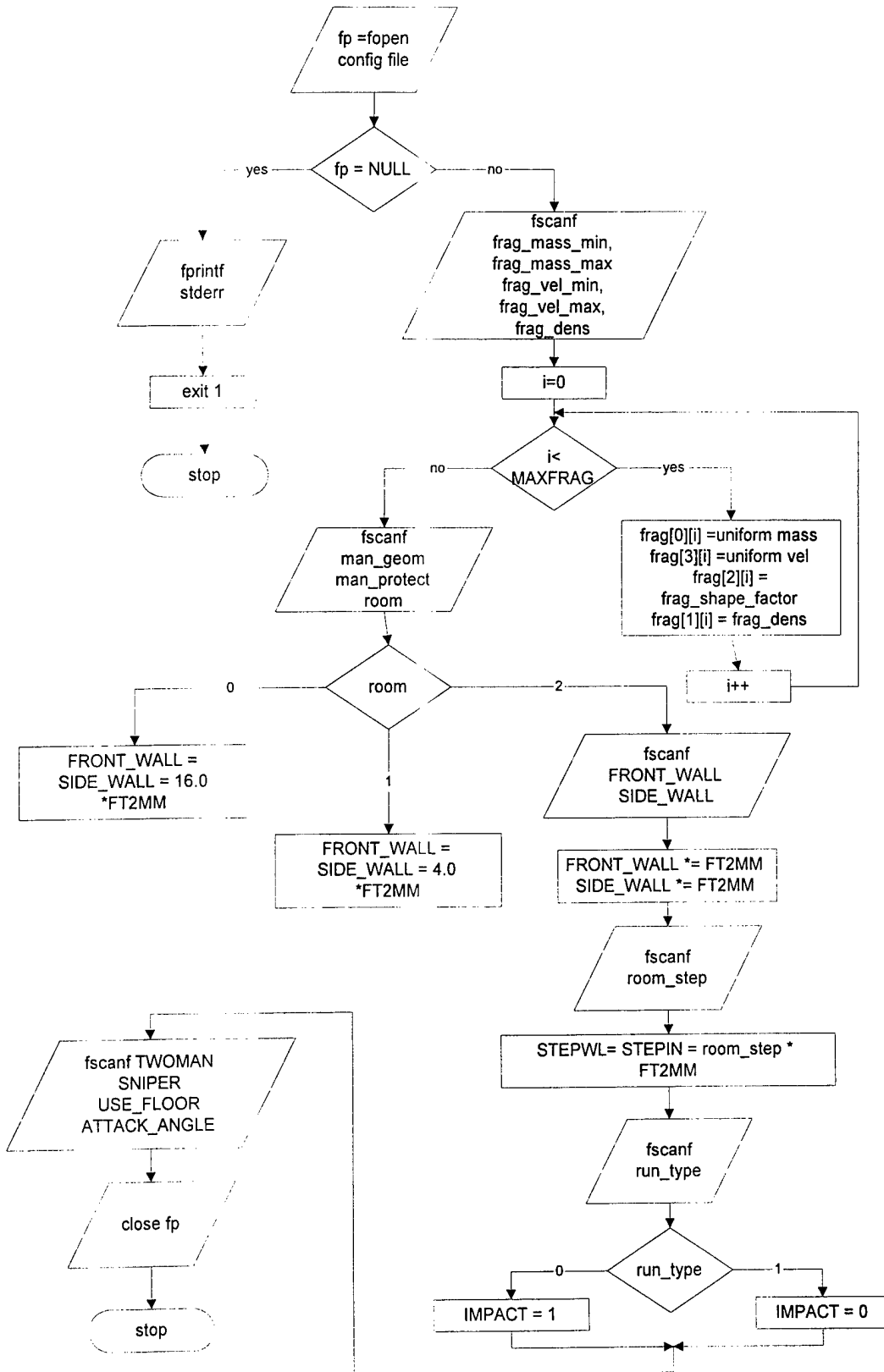
3DpimmsMan VV&A Schedule

Approved V&V Plan	AUG 1, 1999
3DpimmsMan V&V Efforts	SEPT 1, 1999
Draft V&V Report	OCT 1, 1999
Final V&V Report	DEC 1, 1999
Accreditation Plan	FEB 1, 2000
Draft VV&A Report	APR 1, 2000
Final VV&A Report	MAY 1, 2000

DRAFT

Flow Diagram read_conf

DRAFT



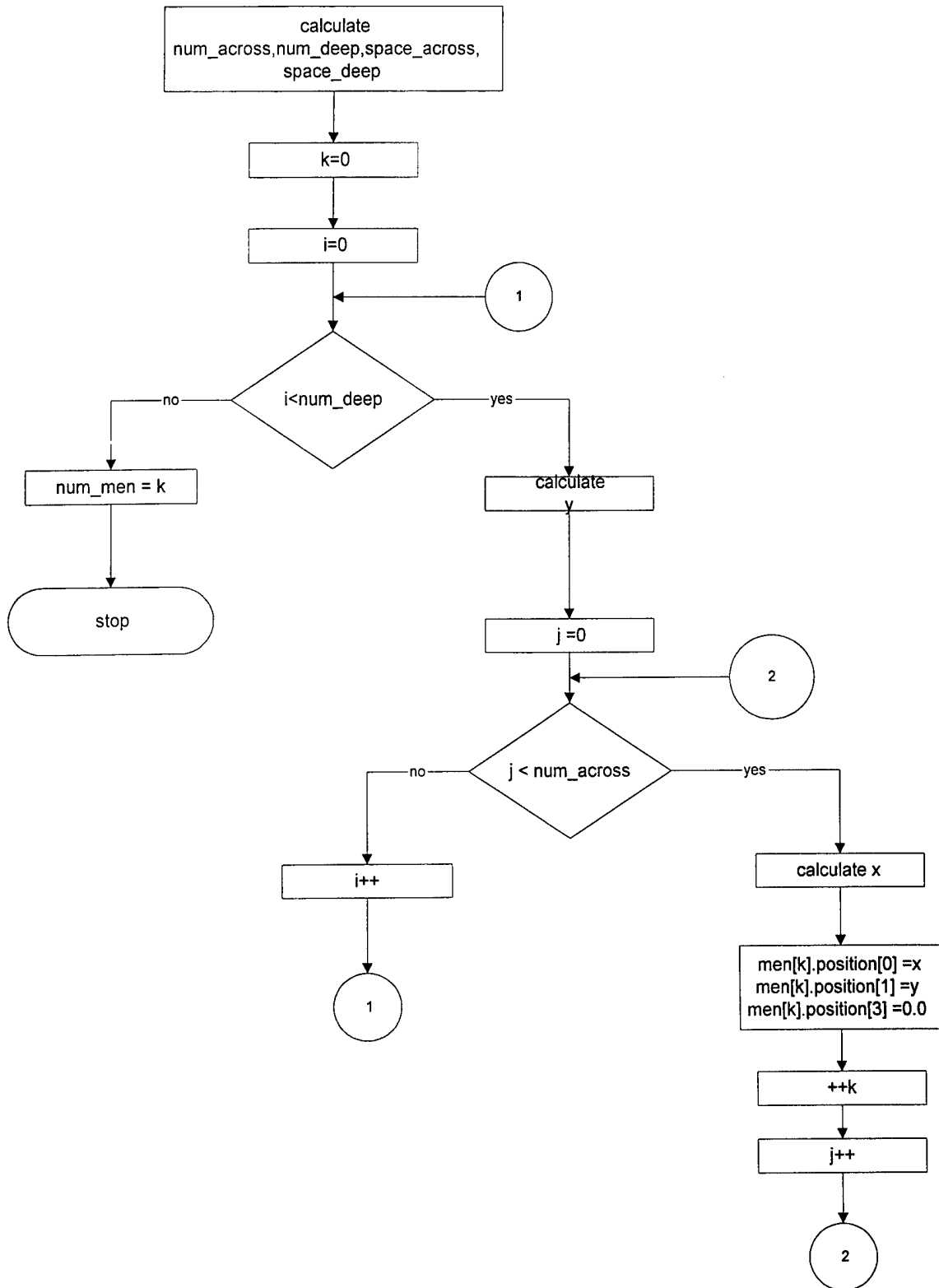
DRAFT

read_conf

Flow Diagram Calc_man_positions

DRAFT

DRAFT



Calculate Man Positions

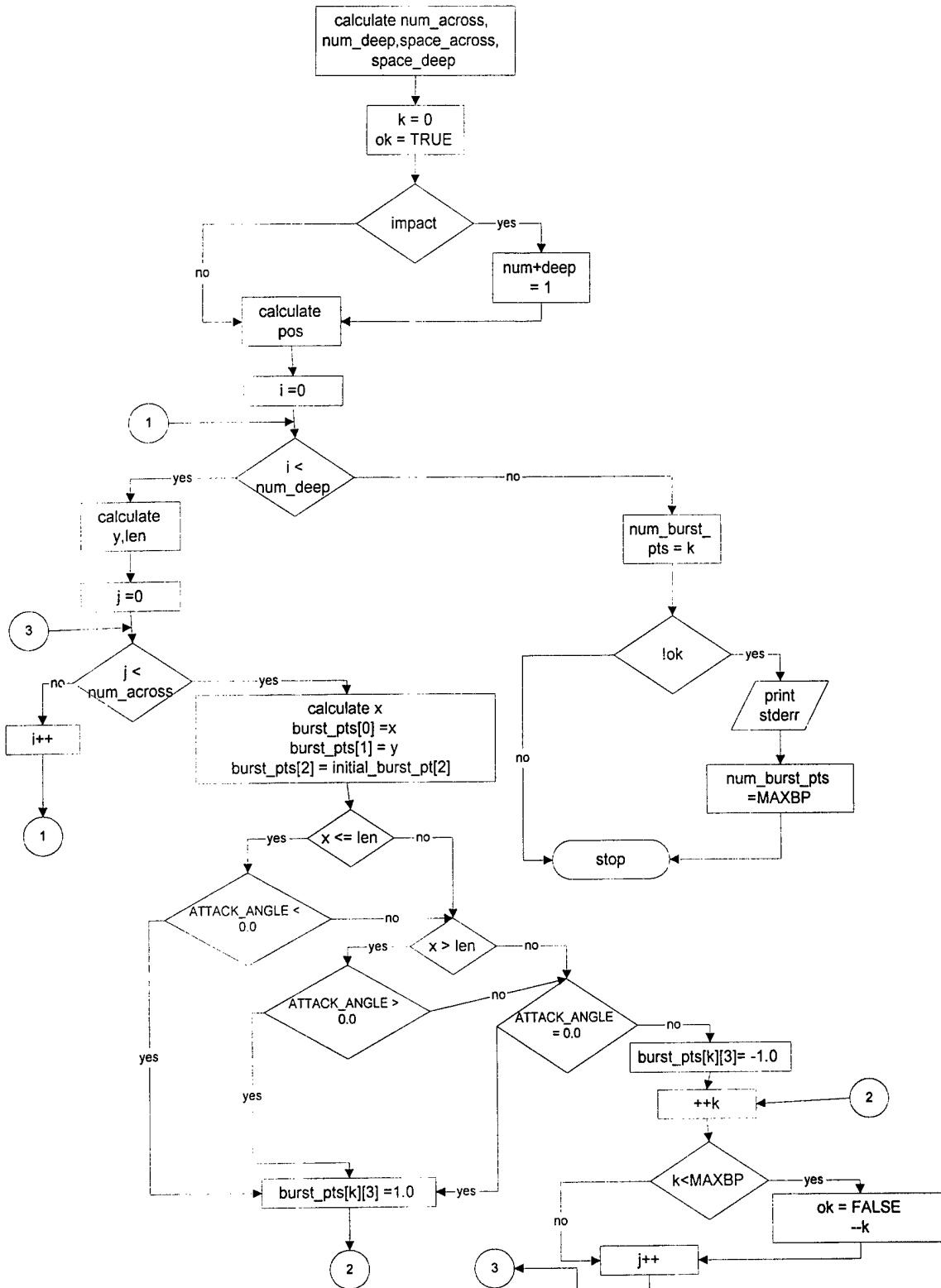
DRAFT

DRAFT

**Flow Diagram
calc_bp_positions**

DRAFT

DRAFT



Calculate Burst Point Positions

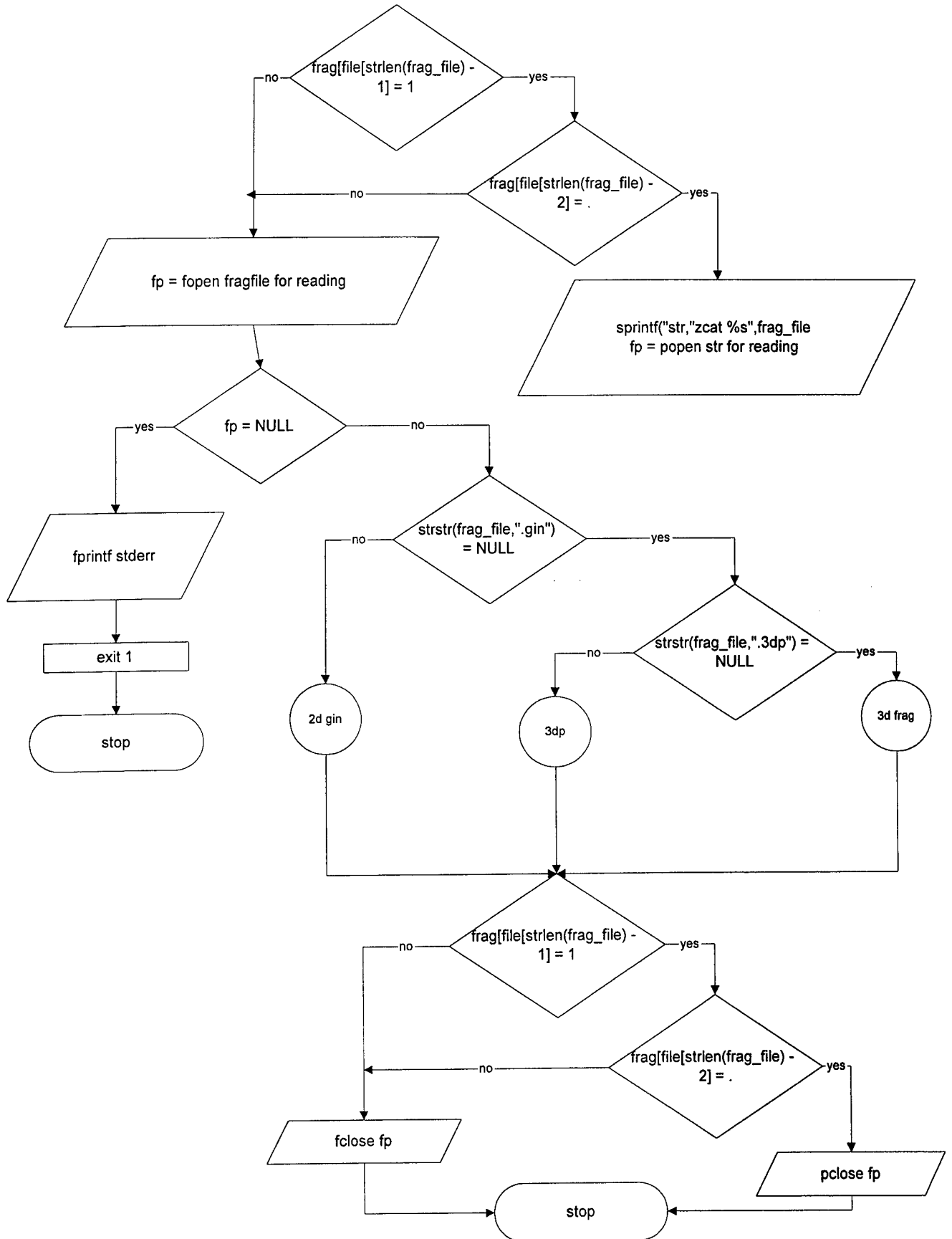
DRAFT

DRAFT

Flow Diagrams
Load_frag

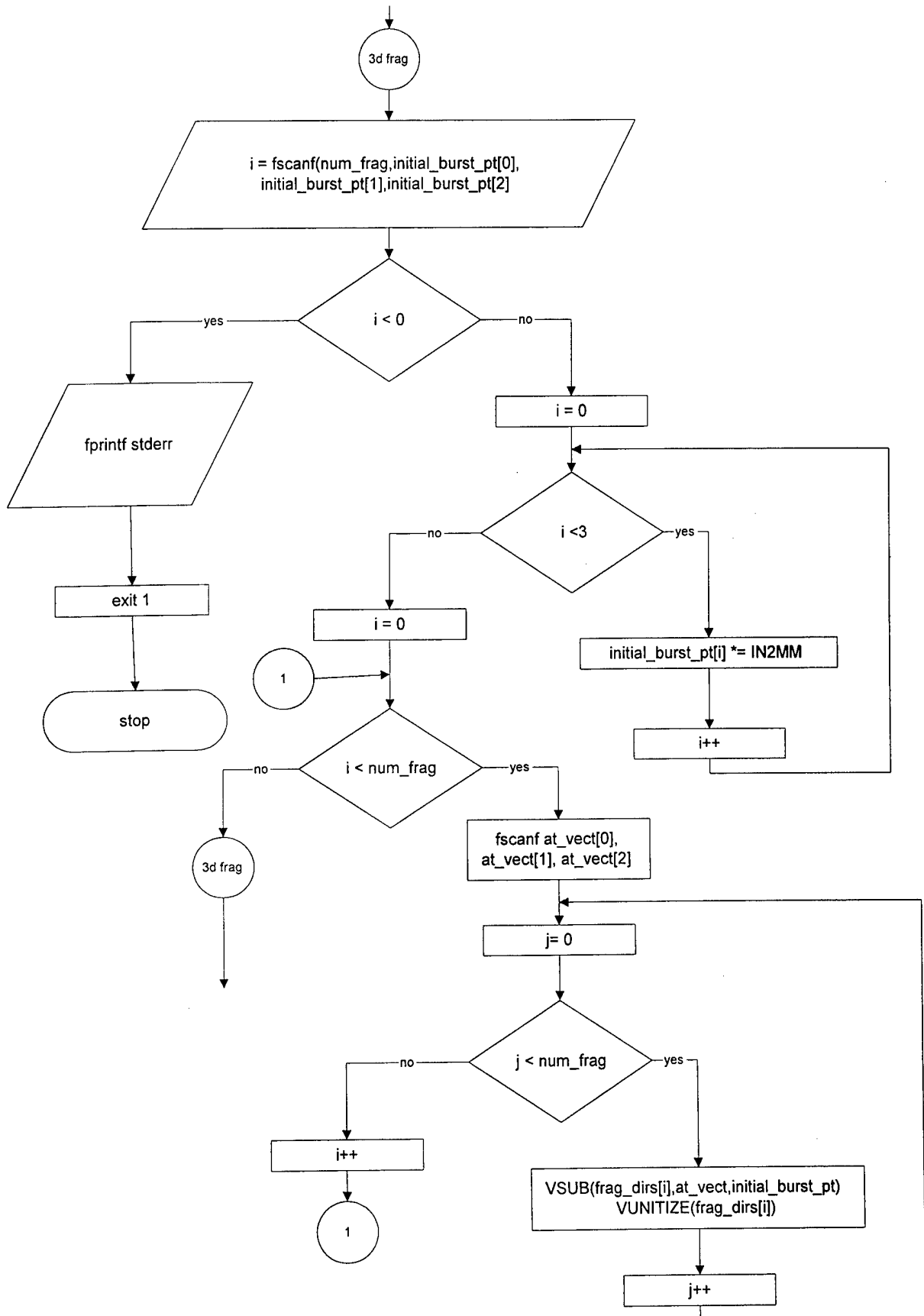
DRAFT

DRAFT



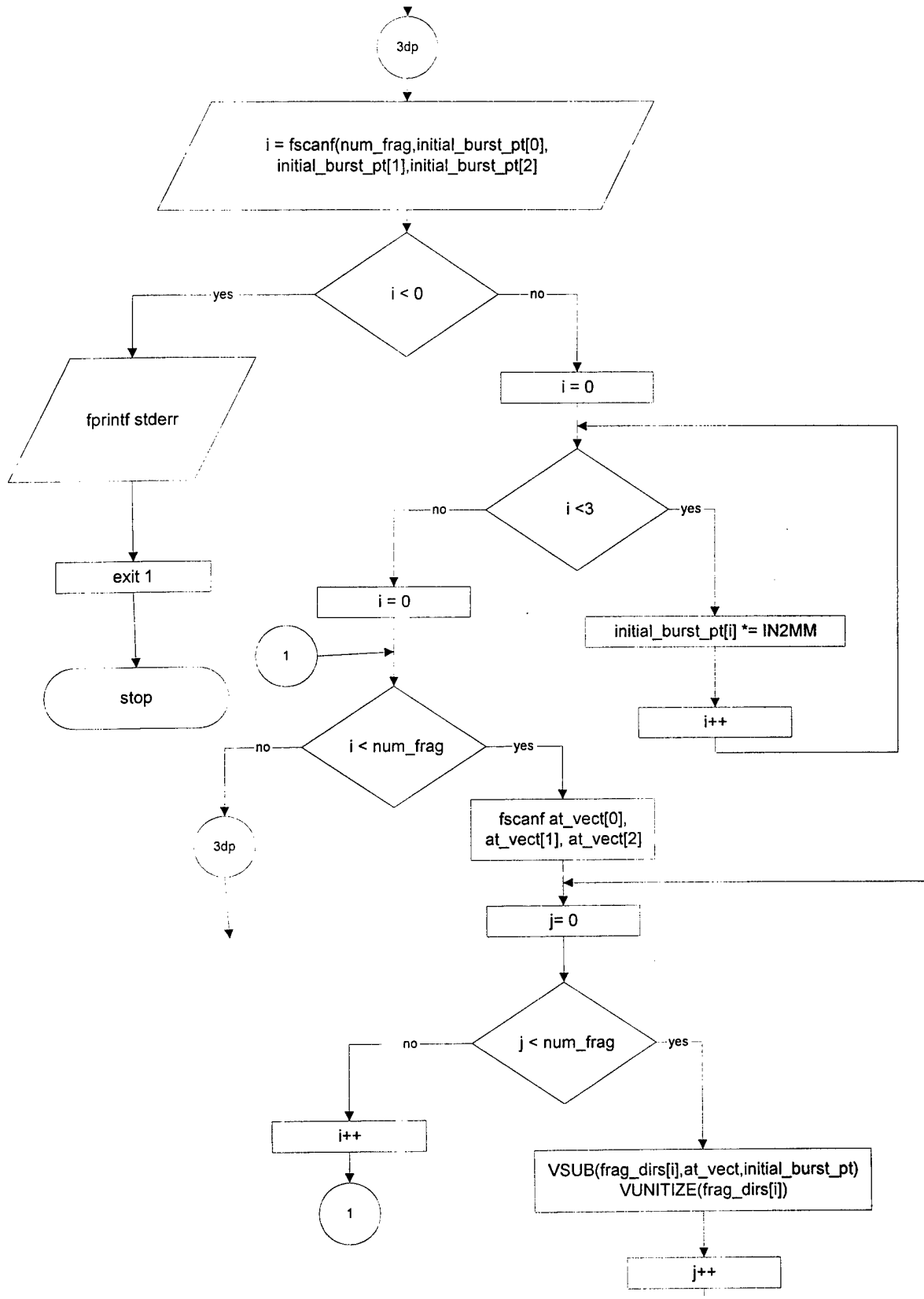
DRAFT

DRAFT



DRAFT

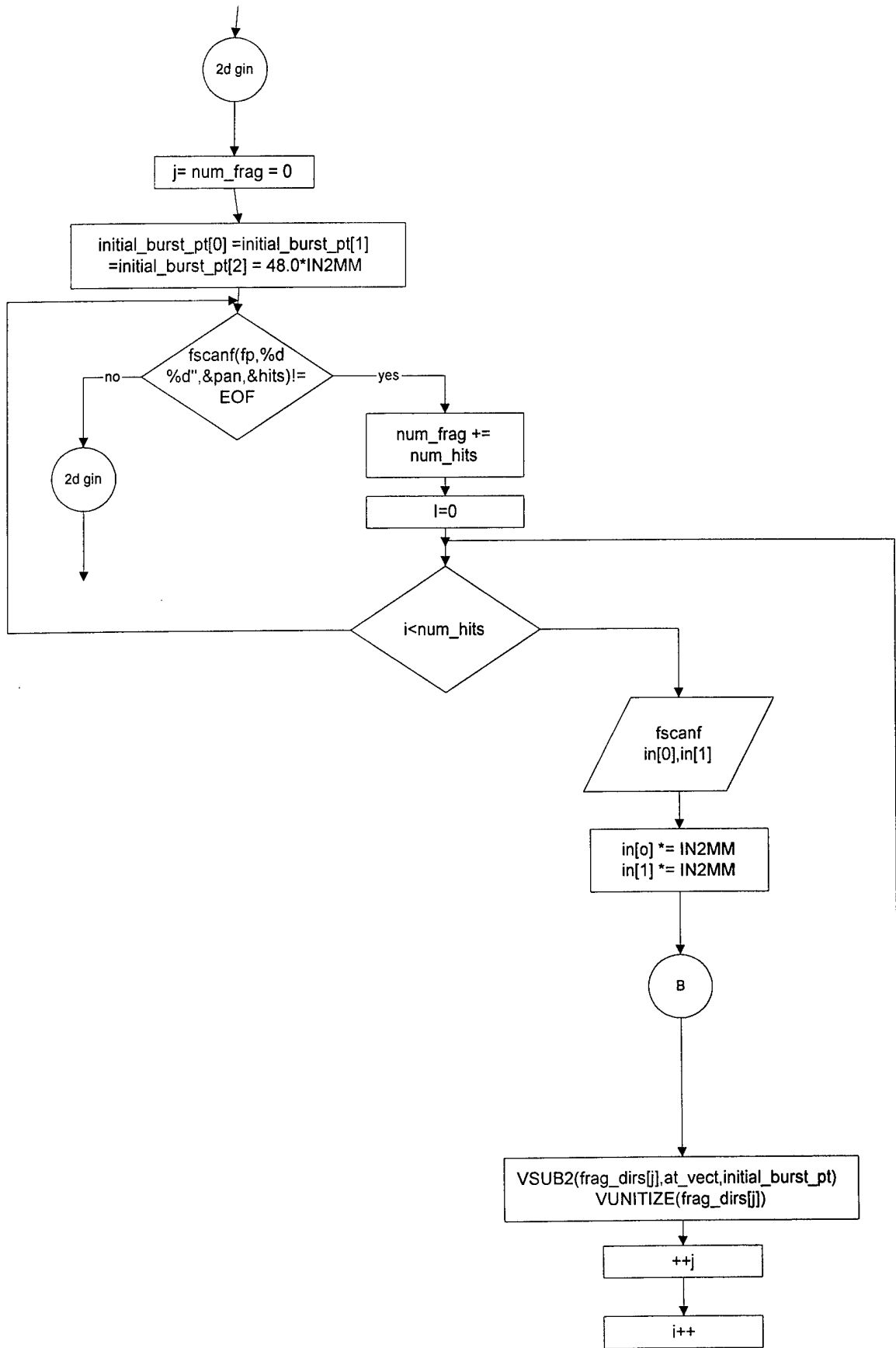
DRAFT

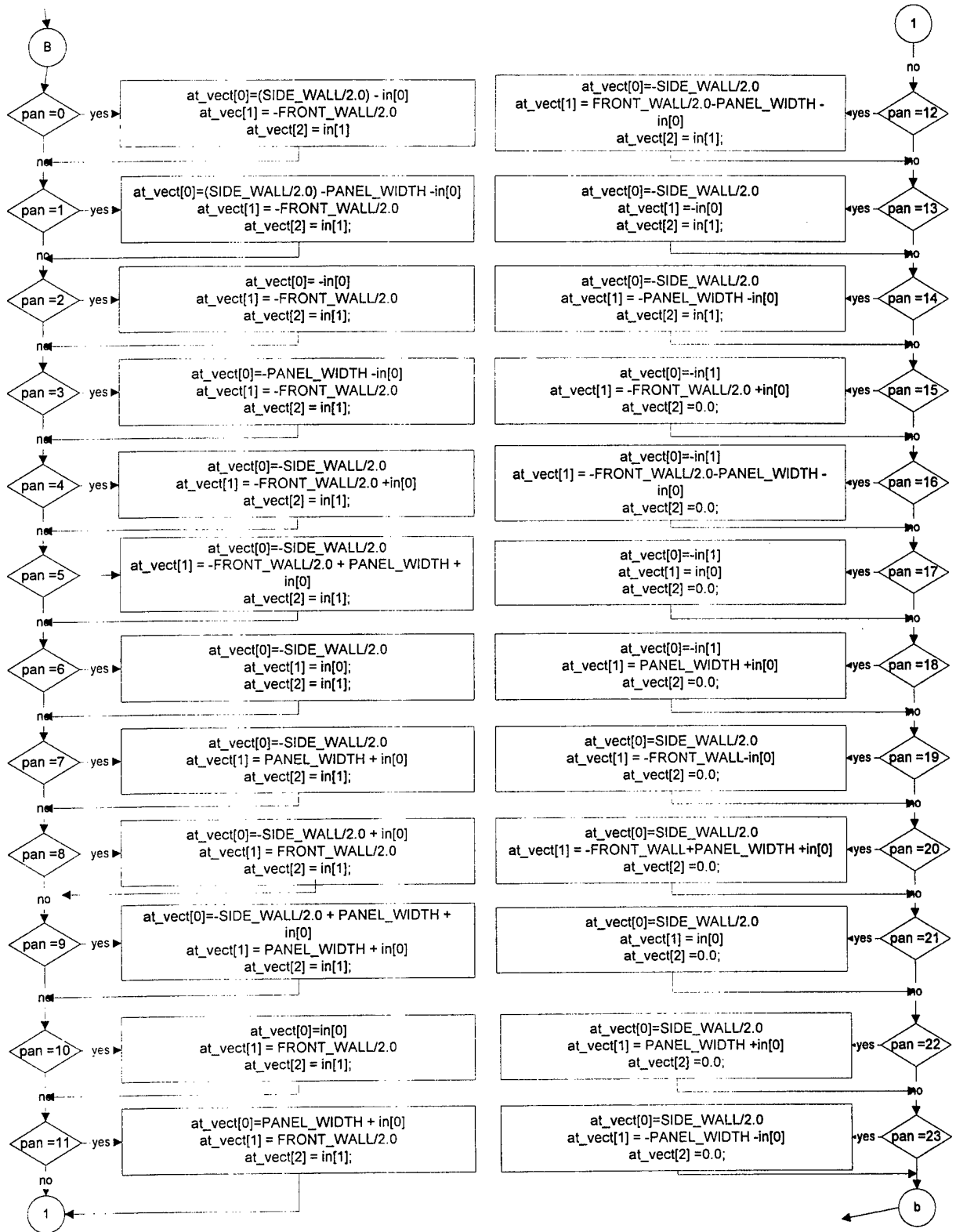


DRAFT

DRAFT

DRAFT





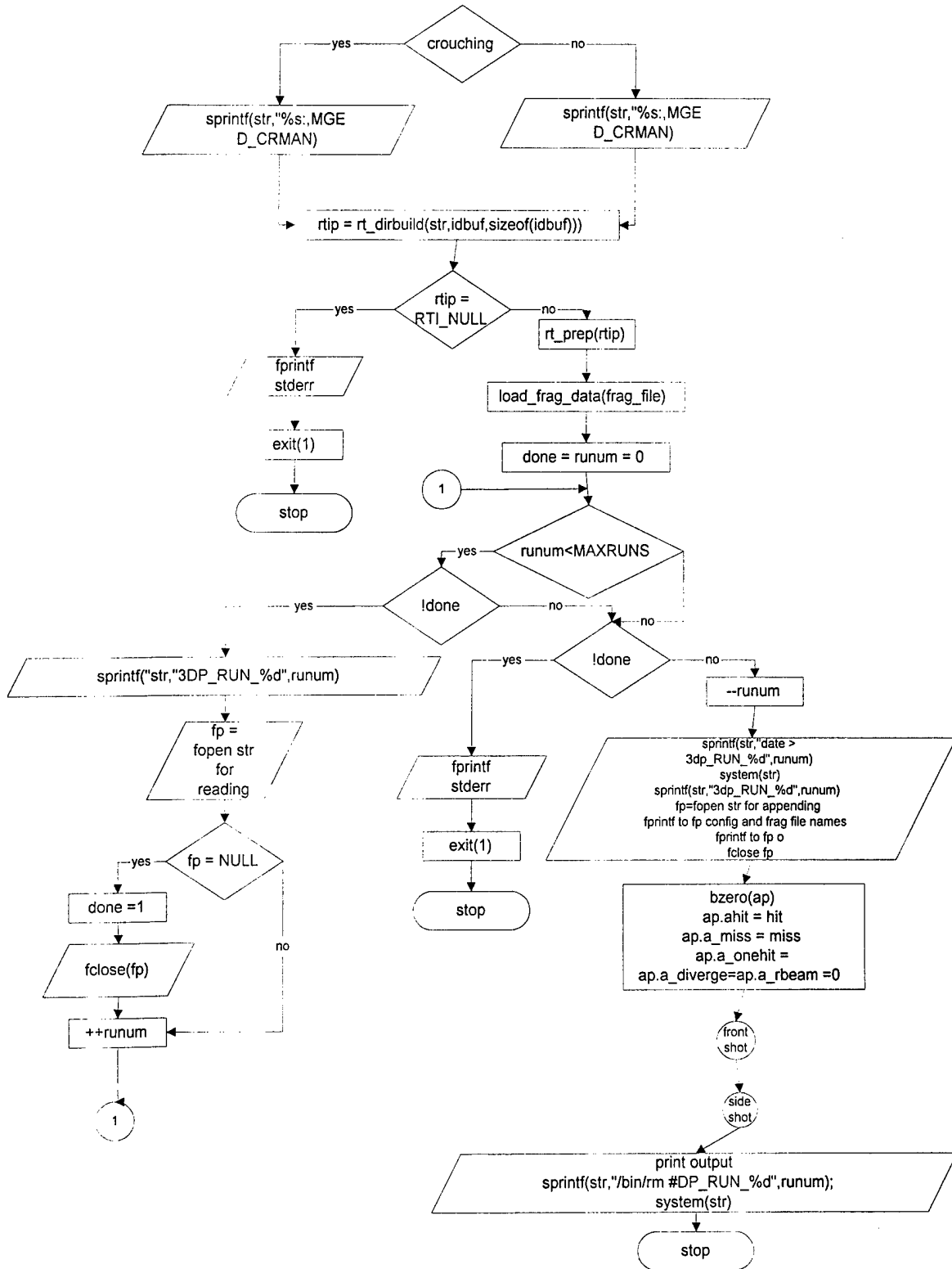
Load Frag

DRAFT

Flow Diagrams Incapacitaion_calc

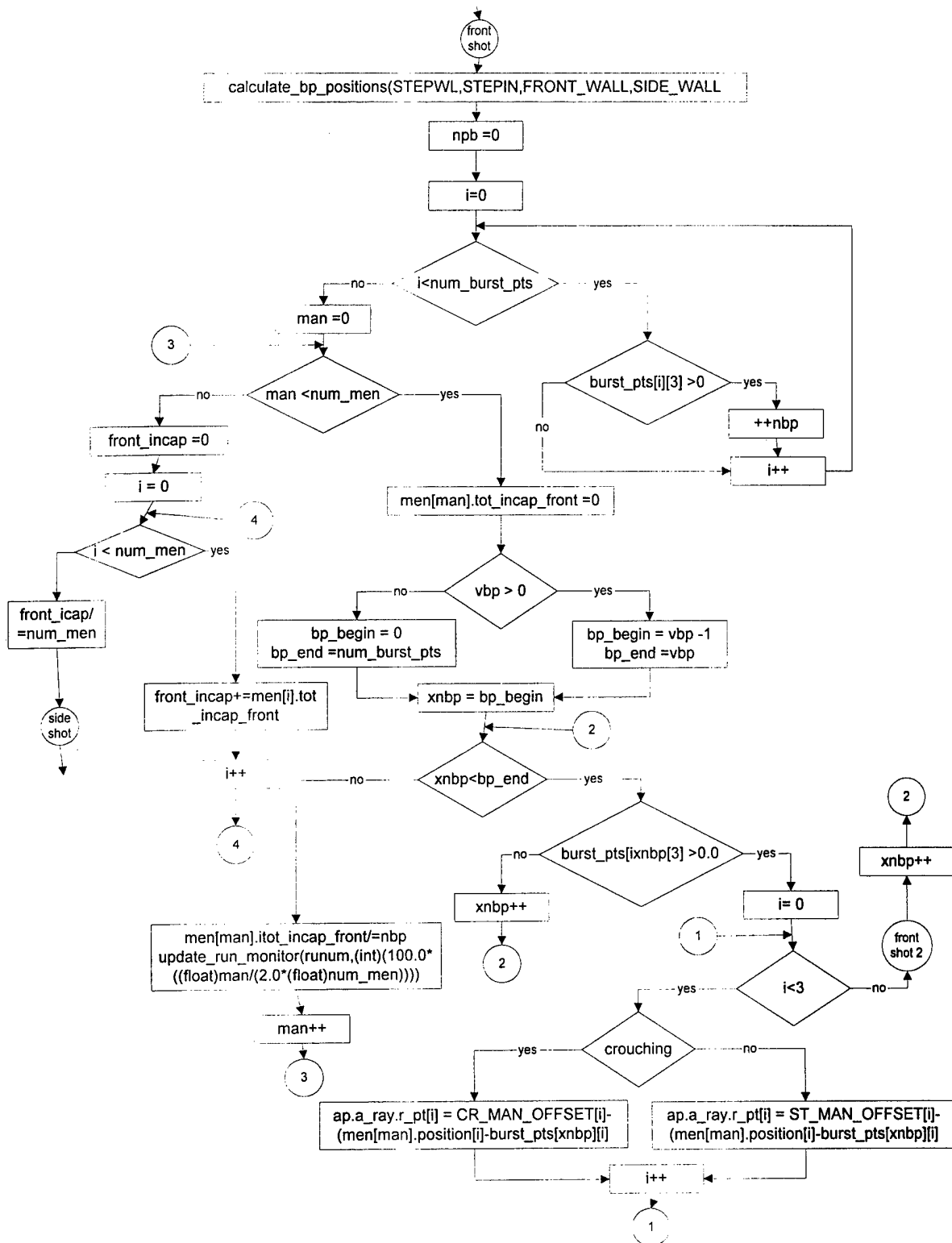
DRAFT

DRAFT



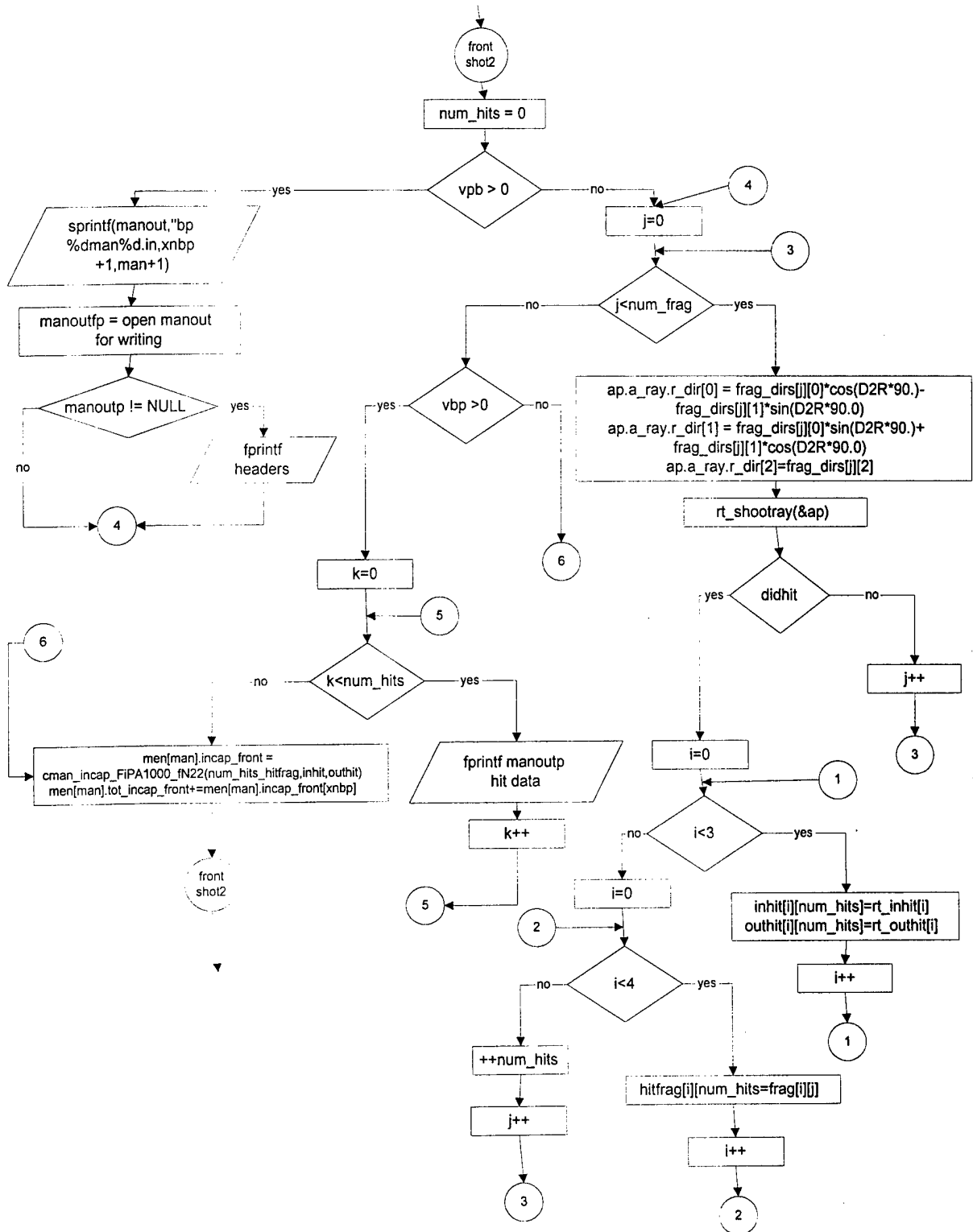
DRAFT

DRAFT



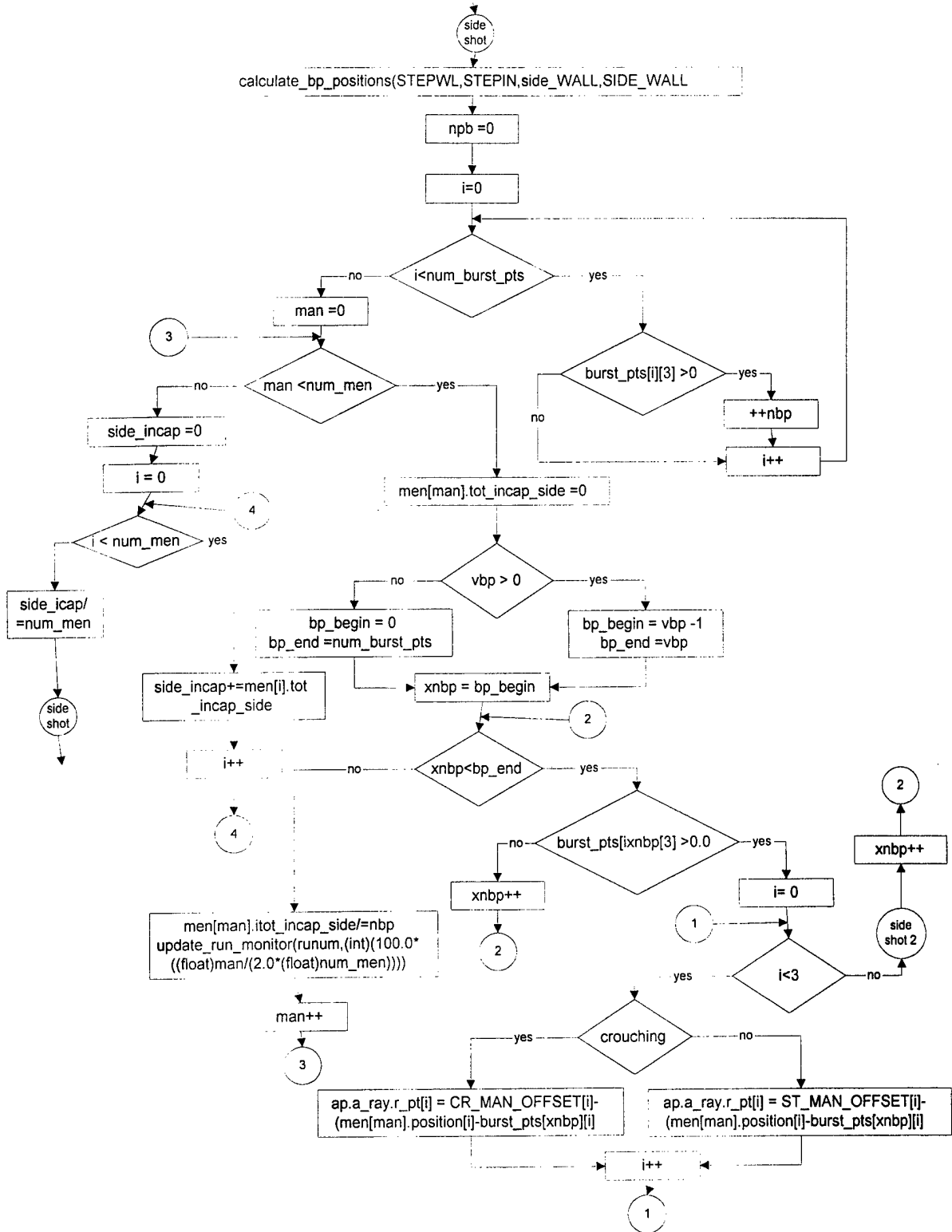
DRAFT

DRAFT



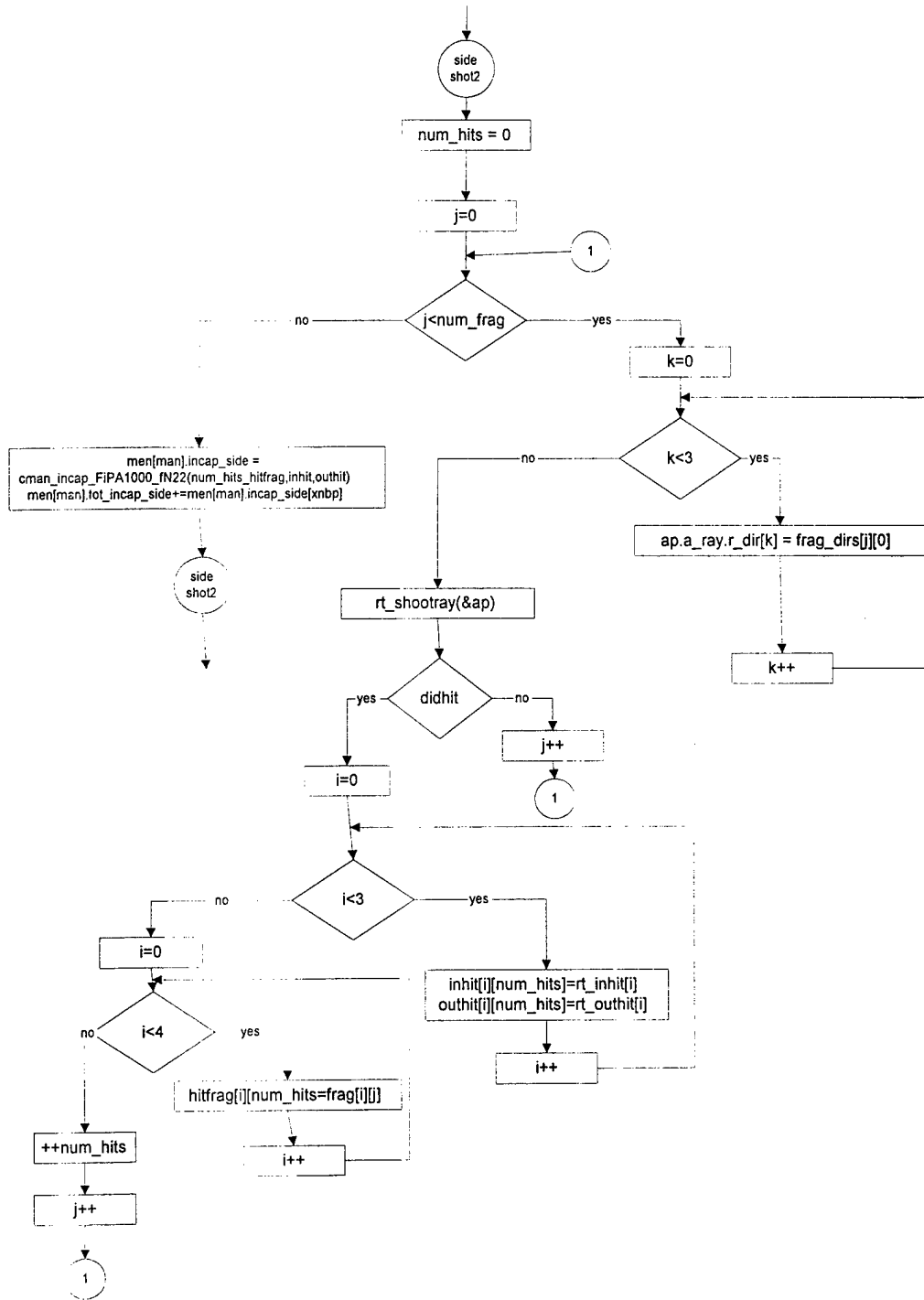
DRAFT

DRAFT



DRAFT

DRAFT



Incapacitation Calculation

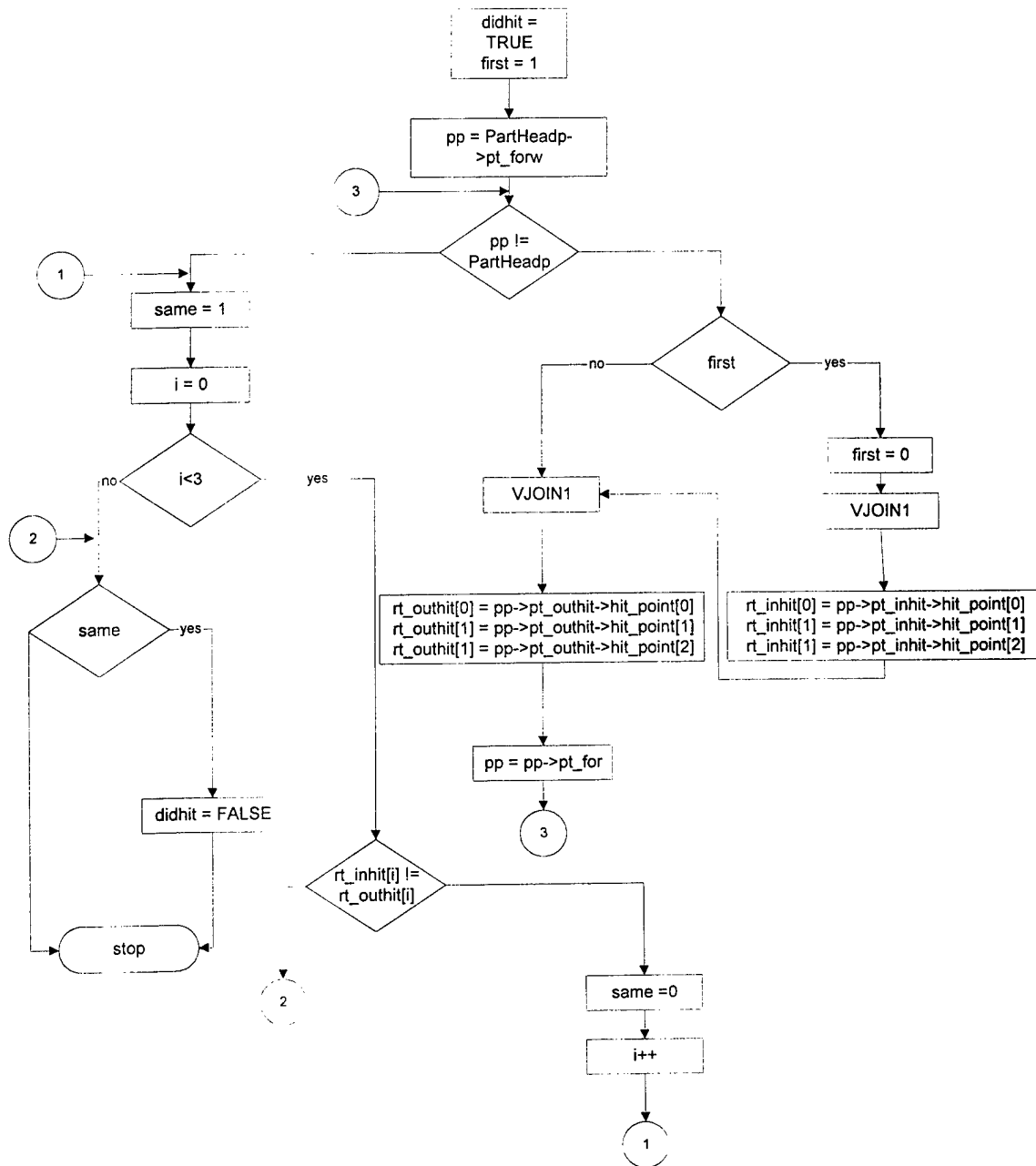
DRAFT

DRAFT

Flow Diagram Hit

DRAFT

DRAFT



Hit

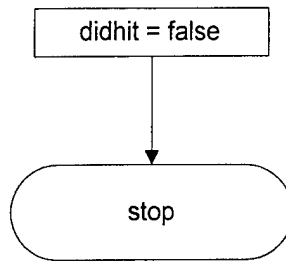
DRAFT

DRAFT

Flow Diagram Miss

DRAFT

DRAFT



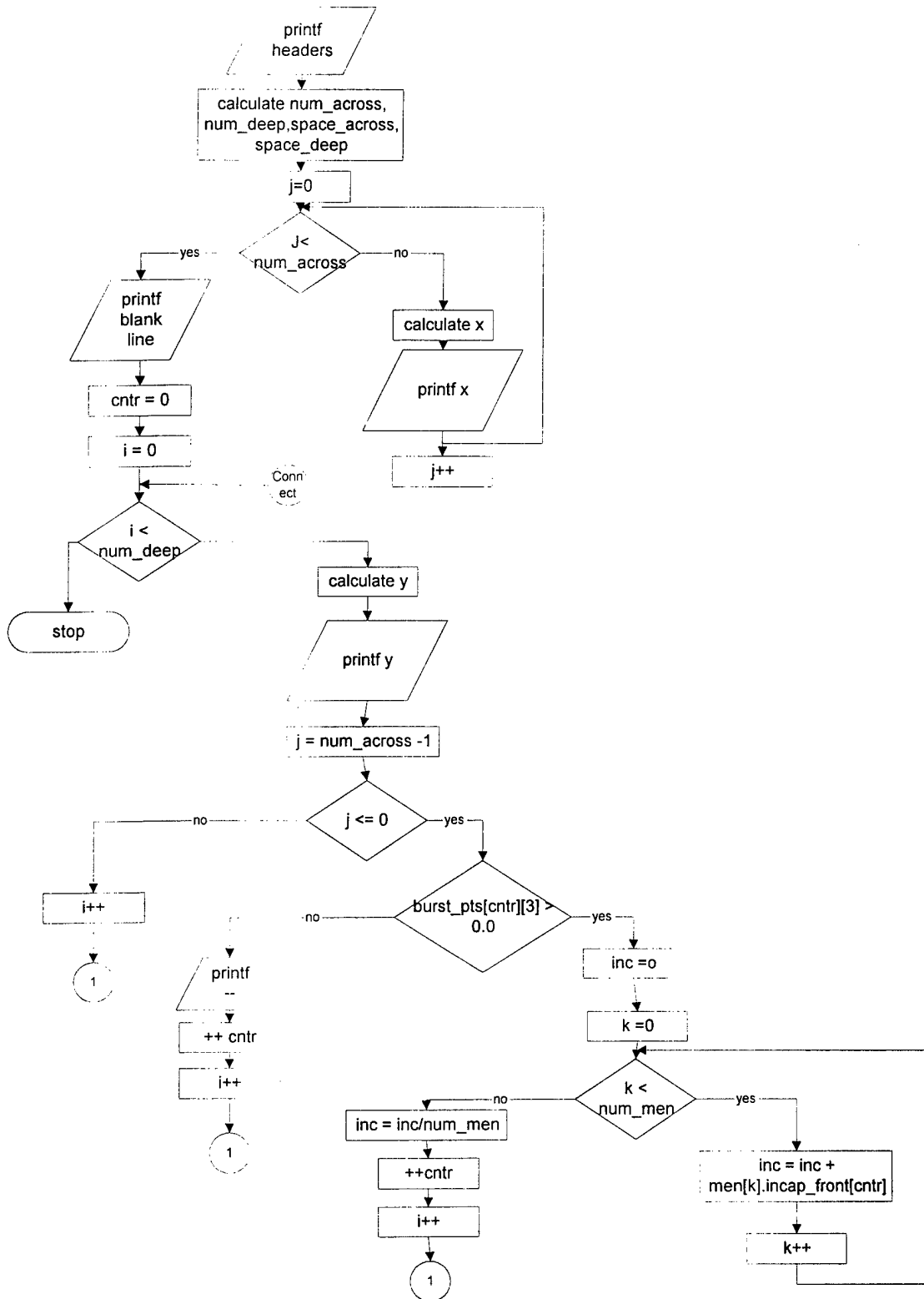
Miss

DRAFT

DRAFT

Flow Diagram
Print Front Matrix

DRAFT



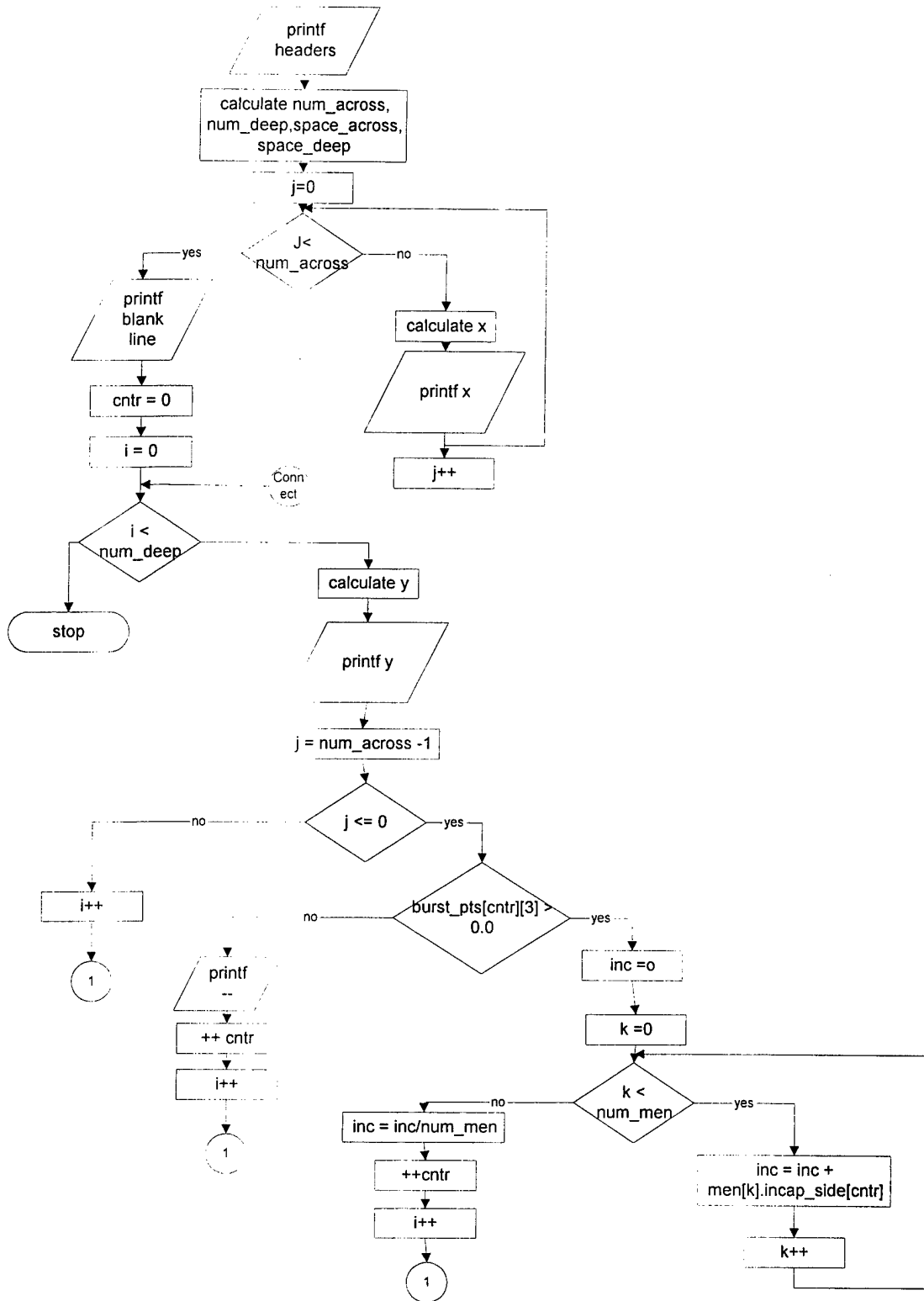
Print Front Matrix

DRAFT

Flow Diagram
Print Side Matrix

DRAFT

DRAFT



Print Side Matrix

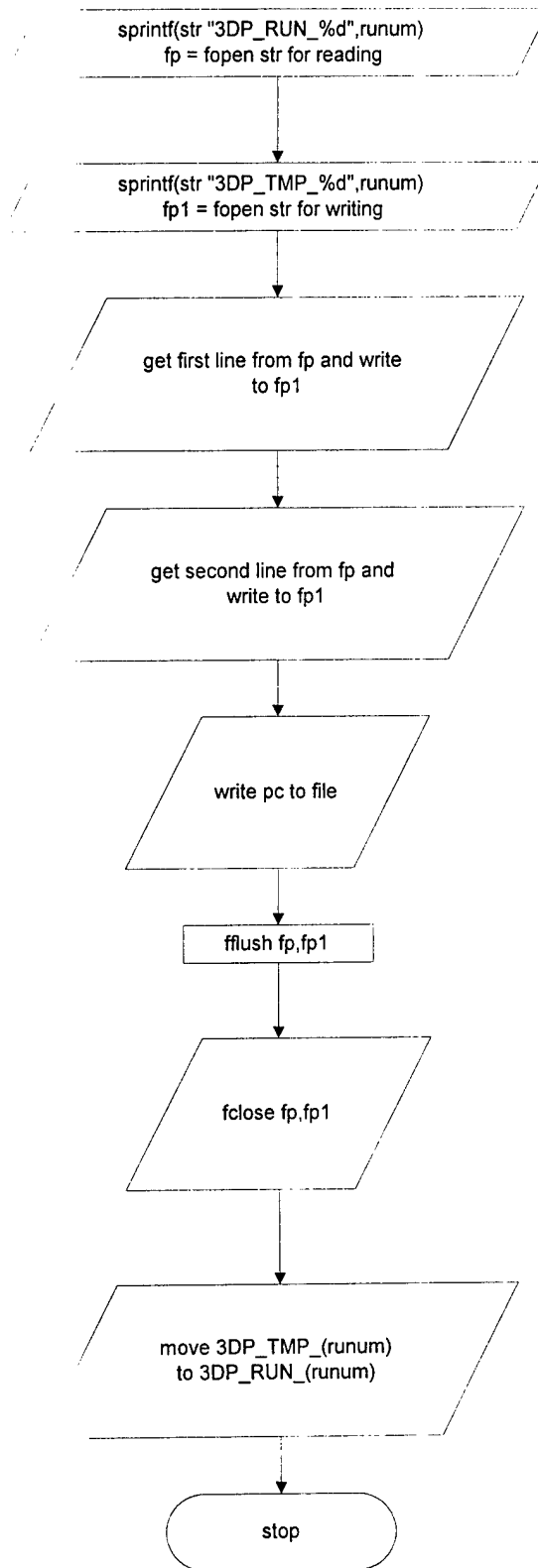
DRAFT

DRAFT

Flow Diagram
update_run_monitor

DRAFT

DRAFT



Update Run Monitor

DRAFT

APPENDIX II - V&V Draft Report

DRAFT

Verification & Validation Report

Executive Summary

The 3DP code is a major enhancement of the legacy codes PIMMS and PICODE. In place of a key assumption in the legacy code computations a direct high resolution calculation of incapacitation has been introduced. This conceptual model and the delta from the PIMMS and PICODE versions will be discussed in the next section. Pimms in this report will be designated 2DPimms and the 3DP code will be referred to as 3Dpimms.

A listing of the total objectives for VV&A can be found in the V&V plan. However, the verification process can be looked at in three parts. First, have the general portions of the legacy models been represented and coded in an appropriate manner? This area concerns the definitions of Burst Points, Aim Points, Man locations, and how these are calculated. This area also defined the calculation of an average incapacitation value which is the average of the men incapacitated for each burst point divided by the number of men in the room.

The second area is the definition and computation of coordinate space for the room and panels in a three dimensional coordinate frame and the fragment path data which is converted from panel space to room space. Also, does the raytrace code provide the appropriate entry and exit wound coordinates for input to the high resolution incapacitation calculation?

The third area is the implementation of the batch mode version of the ComputerMan. Given an entry and exit point of a wound and or wounds, does the incapacitation result match the result that the GUI version of ComputerMan would create and is it reasonable?

This document will outline the conceptual model for both the 2D approach and the 3D approach. The differences will serve to show that the 3D model is a more accurate methodology to calculate this data. It shall also serve as a baseline to understand where the 3D method inherits items like aimpoints, burstpoints, sniper positions, and other legacy concepts.

Conceptual Model Description

This section will attempt to cover the definitions and assumptions that are pertinent to both the 2D Pimms and the 3Dpimms conceptual models. It is not intended to be a comprehensive reference for a more complete description refer to CSC-TR-81-C-0006. The assumptions that are inherent will be covered so that the verification and validation data can be reviewed in this framework.

DRAFT

2dPimms

Methodology

According to the Pimms VV&A plan the methodology of 2D Pimms is as follows,

"The room is filled with men for each attack. A cylinder of specified diameter and vertical cross-sectional area is used to represent a man. A sequence of burst points at a fixed interval along the wall is evaluated in a wall attack using impact fuzed munitions. A matrix of burst points covering the room (in the horizontal plane) is evaluated when delay fuzed munitions are used in a wall attack. Attack orientation or direction relative to the wall is an input value. Fragment data about the burst point are input in five-degree zones relative to the attack or shot direction. For each munition burst point, an incapacitation determination is made for each man in the room. A man is considered incapacitated when struck by one or more lethal fragments. When this criterion is not satisfied, the man is undamaged by the burst. Both the number of incapacitated men and their room locations are stored for each munition burst point. Attacks on each wall are evaluated."

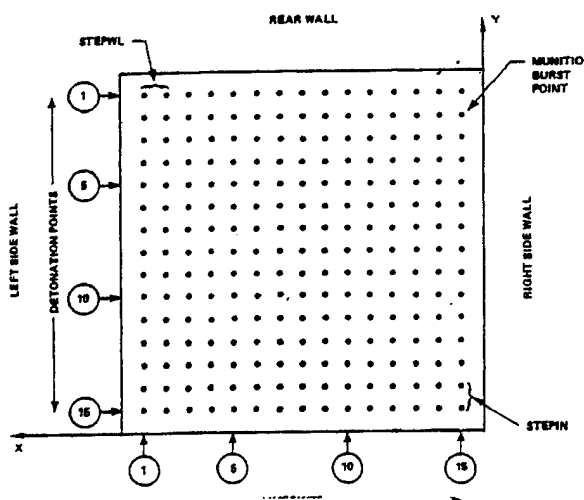


Figure 1 2D Room Burst Points

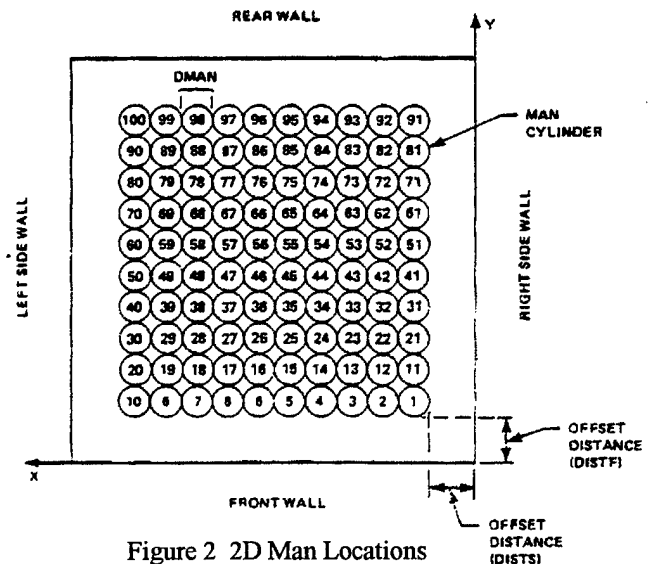


Figure 2 2D Man Locations

Figure 1 & 2 show graphically some of these concepts.

Assumptions Inherent in 2Dpimms

The 2D method of calculating probability of Incapacitation is based upon several assumptions. These assumptions were made in the community due to time, computer and other resource constraints. The first is the definition of an incapacitating fragment. In the 2D methodology an incapacitation fragment is a fragment that in testing completely perforates the appropriate thickness of plywood and celotex. These values were set

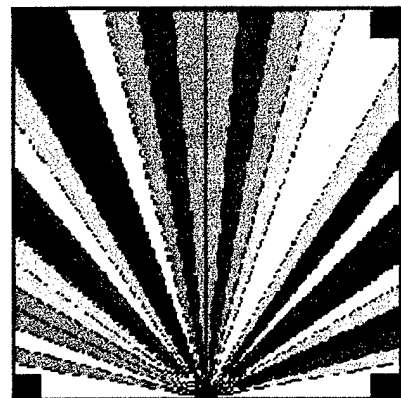


Figure 3 - Fragment Zones

DRAFT

to be $3/8''$ and $1/2''$ respectively. The second assumption is that fragments are uniformly distributed in the quasi-steradian. Figure 3 is a graphical depiction of the zone boundaries for an impact fuze case. These zones are used in 2D to calculate a density per steradian which is used to assign incapacitation to a man position. The third assumption is that probability of being hit by one incapacitating fragment is equal to probability of Incapacitation. Figure 4 shows a panel which has been perforated by fragments meeting this criteria. Previous V&V work has only addressed the input methods to this legacy Fortran model. However, the assumption and techniques have been used as standards in computing Probability of Incapacitation due to fragments.



Figure 4 – Panel Perforations

Figure 5 shows the angle definitions utilized in 2Dpimms to calculate the amount of the presented area of the man who is in each zone and allow the calculation of whether the man cylinder has been hit by the statistical equivalent of one fragment. One interesting assumption in the 2Dpimms methods is the definition of an in-feasible burst point. This concept was coded in the original Pinfib code and carried over into the Pimms implementation. The infeasible burst points are created by a delay fuze munition which impacts a wall at an angle. The angle makes certain burst points in the pre-calculated burst point array, in-feasible. This approach was used because there was insufficient funds for the code developers to write new code, according to the user manual. This assumption has been carried forward to 3Dpimms but a more correct implementation would recompute a new set of detonation points.

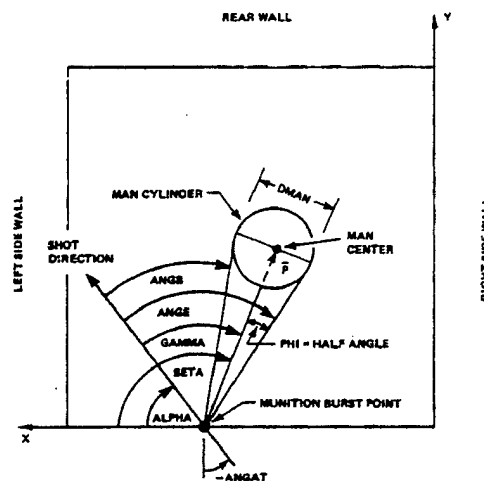


Figure 5 – Angle Definitions in Pimms

DRAFT

3dPimms

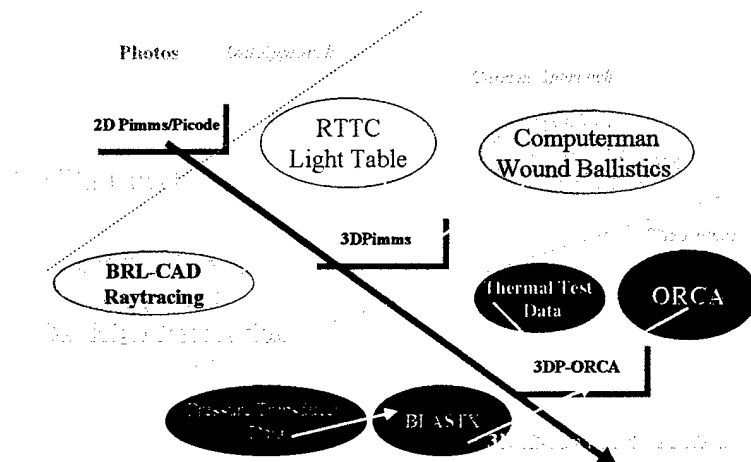
Methodology

The 3D methodology uses the same test data that is collected for the 2D methodology. This data only provides fragments which can perforate the required thickness of plywood and celotex. However, instead of treating the panels as being in a zone and calculating a statistical value (density per steradian) the knowledge of where a panel is in the test setup allows a calculation of the 3D point at which the fragment perforates the wall in the test room. Since the burst point of the munition is known from test or can be assumed, the path that the fragment or debris traveled to get to the perforation can be computed. This path can be transformed into a direction vector. It is this set of vectors that represents the true fragment dispersion pattern from test.

The next step is to find intersections of these fragment paths with the men or personnel who occupy the men/man positions. In the 3DPimms case a crouching man was created to allow for the determination of hit. The three Dimensional method could stop at this point and utilize the assumption of incapacitation from the 2D approach and know with certainty that a man location was impacted by an "incapacitation fragment". This code was utilized as an intermediate code until the linkage to the High resolution wound ballistics model could be produced.

The linkage to the ComputerMan software was accomplished quickly and properly because ComputerMan was written in C++ and was modular in design. This allowed a modification of their current batch mode code to become the method of taking multiple wound paths and computing an incapacitation value for that person.

The conceptual model can now be viewed in the three parts that make up this Verification and Validation report. The first part is the definition and the creation of burst points, man positions and a "room" for simulation activity to be computed. Secondly, the test data is transformed into a three dimensional point, then into an appropriate direction vector so that a raytrace from the burst point to the impact point can determine if a man was hit and



DRAFT

DRAFT

where was he hit in our three dimensional room. Finally, now that the man was hit by one or several fragments is an appropriate calculation done to determine incapacitation.

An assumption that was needed to make this happen is that mass and velocity data can be input and then distributed to the fragments which were collected from test as perforating the plywood panels. Currently, the analyst inputs into the configuration panel a minimum and maximum value in both mass and velocity. The 3dp program utilizes a uniform distribution function to specify mass and velocity to a particular fragment. Figure 6 shows the configuration screen and these inputs.

FRAGMENT SETUP			
	Minimum	Maximum	
MASS: (grams)	4.0	6.0	
	Minimum	Maximum	
VELOCITY: (m/sec)	1250.5	2200.8	
SHAPEFAC:	1.5		
DENSITY: (gm/cc)	16.1		

MAN SETUP	
Man Geometry	<input checked="" type="checkbox"/> STANDING
	<input checked="" type="checkbox"/> CROUCHING
Man Protection	<input checked="" type="checkbox"/> NONE
	<input checked="" type="checkbox"/> VEST
	<input checked="" type="checkbox"/> BOTH

ROOM SETUP	
ROOM SIZE	
<input checked="" type="checkbox"/> 16 X 16 (Ft.)	
<input checked="" type="checkbox"/> 4 X 4 (Ft.)	
<input checked="" type="checkbox"/> User Defined	
WALL LENGTH	16
WALL WIDTH	16
WALL STEP	1
IMP. ANGLE	0.0
FUZING TYPE	
<input checked="" type="checkbox"/> IMPACT	
<input checked="" type="checkbox"/> DELAY	
EXTRA OUTPUT	
<input type="checkbox"/> TWO MEN	
<input type="checkbox"/> SNIPER	
FLOOR FRAGMENTS?	
<input checked="" type="checkbox"/>	

Figure 6 – The configuration setup screen

Additionally, a fragment shape factor is set and the density of the fragments is set for the grenade based upon the material type. If, grenade designs specify multiple materials in the grenade fragmentation pattern the analyst would have to decide the effect of this assumption. However, since test data does not tell us the makeup of the fragments that actually perforate the panels, hydrocode results or other tests would have to be utilized to accurately set this value.

Verification and Validation Results

In an effort to make this report usefull and easy to trace back to the Verification and Validation Plan , Table 1 shows a restatement of the 9 V&V Plan requirements and where these are addressed in this report.

DRAFT

V&V Plan Requirement Item	Section Addressing Requirement
1) simulation loads fragment data correctly	
2) simulation must calculate man and burst points correctly	
3) simulation must define fragment path and impact point in the Cman coordinate frame	
4) simulation must aggregate results correctly	
5) simulation must match intuitive results	
6) simulation must create proper burst points	
7) deviations from 2D must be explained	
8) simulation must calculate room coordinates from 2D fragments per panel	
9) prove that fragment input to incap model is consistent with fragments from panels	

TABLE 1 V&V Plan to V&V report Crosswalk

Section 1 Legacy methods implemented correctly

The first legacy method is the creation of burst points in the room. The routine that calculates these in 3dp.c is the calc_bp_positions subroutine. This subroutine requires that four values be passed to it. The values that are needed are the room step in mm. The user enters the number in feet and the read_config subroutine translates it to millimeters. Currently, the 3dp.c code sets the stepwl and stepin variables to be equal which means that only square arrays of burst points can be created. This can easily be modified if the test areas begin doing tests against rectangular room. The attack angle is also read from the configuration file and is set globally. The maximum number of burst points is also set in the 3dp.c code in the define section at the top of the code. The code that sets the variable in the burst_pts array in 3dp.c can be seen in Figure 7.

```

/*
 * Calculate burst point locations within the room based on wall sizes and steps.
 * stepwl,stepin,front,side given in mm
 */

calc_bp_positions(stepwl,stepin,front,side)
float front,side,stepwl,stepin;
{
    int num_across,num_deep,i,j,k,ok;
    float space_across,space_deep,x,y,pos,len;

    num_across = front / stepwl;
    num_deep = side / stepin;
    space_across = (front - (float)num_across * stepwl) / 2.0;
    space_deep = (side - (float)num_deep * stepin) / 2.0;
    k=0;
    ok = TRUE;
    if (IMPACT) num_deep = 1;
    pos = ATTACK_ANGLE < 0.0 ? side / 2.0 : -side / 2.0;
    for (i=0;i<num_deep;i++) {
        y = (side / 2.0) - space_deep - (stepin * (float)i) - stepin / 2.0;
    }
}

```


DRAFT

```

len = ATTACK_ANGLE < 0.0 ? pos-(side/2.0-y)*tan((-1.0*ATTACK_ANGLE)*D2R) : pos+(side/2.0-
y)*tan(ATTACK_ANGLE*D2R);
for (j=0;j<num_across;j++) {
    x = (front / 2.0) - space_across - (stepwl * (float)j) - stepwl / 2.0;
    burst_pts[k][0] = x;
    burst_pts[k][1] = y;
    burst_pts[k][2] = initial_burst_pt[2];
    if ((x <= len && ATTACK_ANGLE < 0.0) || (x > len && ATTACK_ANGLE > 0.0) || ATTACK_ANGLE == 0.0)
burst_pts[k][3] = 1.0;
    else burst_pts[k][3] = (-1.0);
    ++k;
    if (k > MAXBP) {
        ok = FALSE;
        -k; } }
    num_burst_pts = k;
    if (lok) {
        fprintf(stderr, "\n** WARNING:: Number of burst points exceeds maximum.\n\nValue set to maximum
(%d).",MAXBP);
        num_burst_pts = MAXBP; }
}

```

Figure 7 – The Calc_bp_positions subroutine

A graphics code which checks the functioning of this routine was written during the checkout of the intermediate code. This code bp.c shows that given the correct inputs in stepwl, stepin, and angle the code properly calculates the burst point locations and whether according to the 2Dpimms method the burst point is feasible. This value is stored in the burst_pts[k][3] location. Figure 8 shows a graphical representation of the calculated values from the calc_bp_positions subroutine.

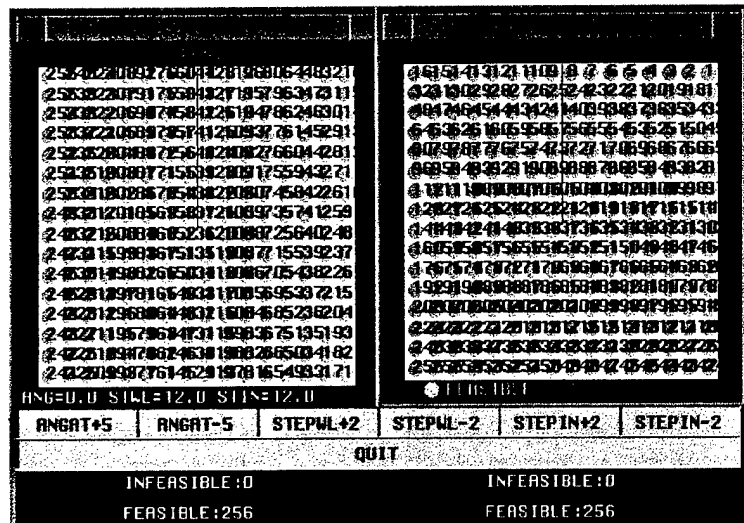


Figure 8 – The Bp.c code for visualizing burst points

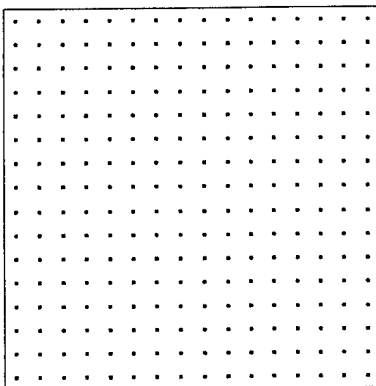


Figure 9 is another graphics of this same subroutine plotted in the 2d coordinate frame with the x and y axis as defined in figure 2. The user is cautioned to refer to the next section to see how the coordinate frame effects the numbering of these burst point locations. Only the numbering is effected not the actual location of the burst point.

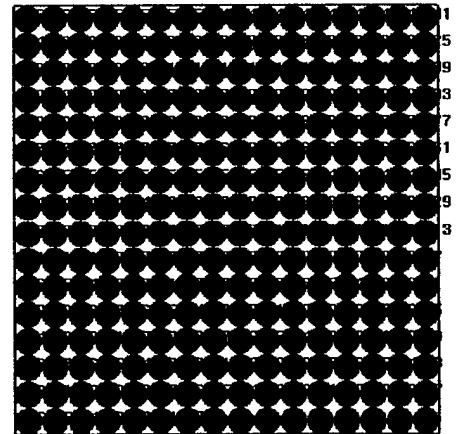


Figure 9 – Two different visualizations in 2D of the burst point locations.

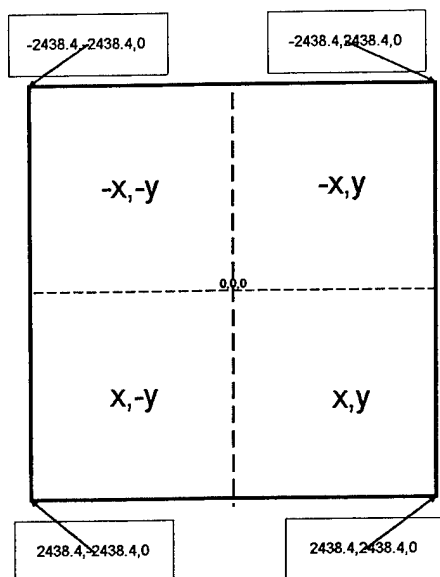


Figure 10 – Coordinate Frame

Output matrix and averaging

A very important part of the legacy techniques which was required to be maintained is the averaging of the burst points. The results for each burst point position for all men in the room is the average of the number of men incapacitated divided by the number of men in the room.

Section 2 Coordinate frame and geometry of man

Coordinate Frame

The Coordinate frame of the 3Dpimms simulation can be seen in Figure 10. This shows that looking down on the room the positive x and y location is in the right lower corner of the room. The corner coordinates are shown in millimeters. In english units, assuming a 16 foot room, the corners are at 96 inches. This is between panel 12 and 13. Figure 11 shows the panel locations for a typical room. The Floor Panels will be covered in their own diagram. With this coordinate system in place a code was written to convert from a 2 Dimensional panel oriented data format to a full 3 Dimensional format. A primary focus of the verification part of this document is the coordinate frame and the conversion of fragments into the proper fragment location in 3D from the test data as collected at the range. Appendix 1 contains a detailed view of test data converted by the code displayed graphically and in plain text. A spreadsheet output is part of Appendix 1. This spreadsheet converts a full grid of “fragments” at one foot increments for each panel into the proper coordinate frame. The equations utilized are identical to those found in the code.

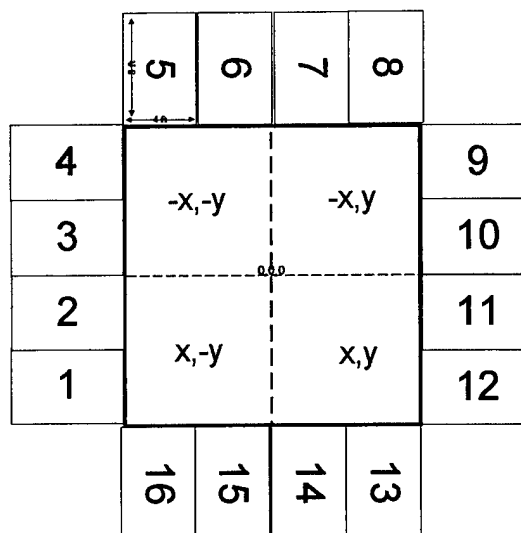
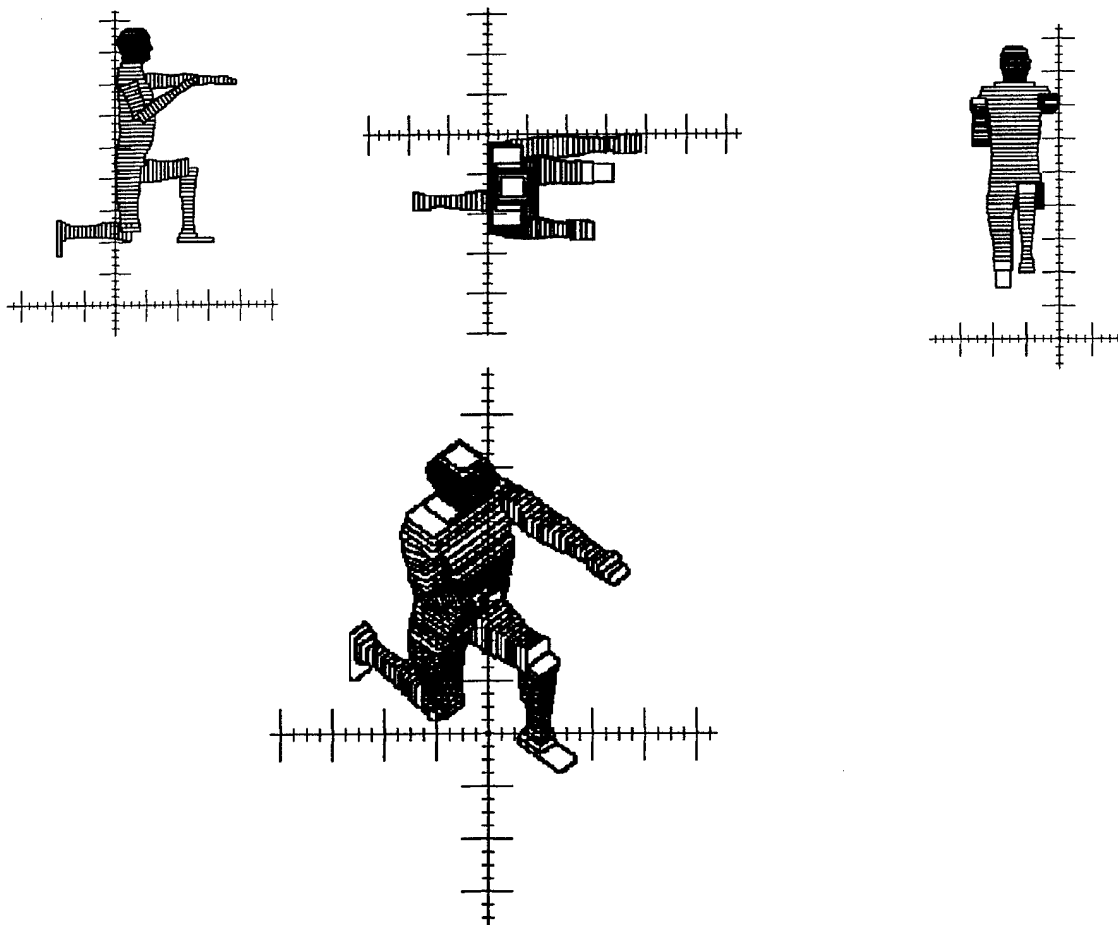


Figure 11 – Panel Locations

DRAFT

ComputerMan Geometry

The geometry was created by evaluating the graphical user interface version of ComputerMan. It was found that the Graphical User Interface used the concept of bounding boxes to specify the exterior of the wound ballistics data. In fact, they utilize the bounding boxes as a way to position shotlines through the actual Anatomical data. The first geometry that was extracted for this same purpose proved to be too large and cumbersome to utilize. It was essentially every skin voxel in the Anatomical description. Appendix II displays the early geometric work and some interesting current work. The bounding box is a much more useful and elegant implementation. The exact coordinates for the standing man were extracted from the ComputerMan code. These sections were then read into BRL-CAD to create a set of solids which represented the bounding sections as ARB8 geometrical solids. These solids were then built into a MAN region. The next step was to achieve the rotations needed to have a crouching man. This was a more difficult task. The rotation angles were available from the code, but BRL-CAD rotates about specific points when editing geometry. Regions were created to match the definitions of lower leg and upper leg. Then rotations about the correct point allowed the BRL-CAD bounding box man to crouch as needed. Comparisons were then made to the Graphical User Interface versions with measurements made on scaled prints to compare all vertices. Figures 12 -15 show different views with a scaled axis. During this V&V



Figures 12 - 15, Right , Top, Front, and 3D view of Bounding Box Crouching Man

DRAFT

effort a computer code was written to take the exact rotation angles and utilize the exact methodology to rotate the bounding boxes as found in ComputerMan. This approach was compared with the original manual rotations to obtain a crouching man. The differences were less than a few millimeters for leg locations. However, this analysis leads to a discussion of the necessary pairing of geometry file and the crouching man offset as set in the 3dp.c file. The crouching man offset is set because the crouching man geometry does not have the 0,0,0 point set at the middle of the geometry at ground level. The crouching man actually floats in the air approximately 1.33 ft above the ground. This is an artifact of the method used to rotate the sections into a crouching man. The torso and head were left at the same point as they exist in the standing man anatomy file, and the legs were bent and rotated in relation to the torso. Therefore, numerically the man floats. In order to place the man at the point required for the 3dp.c methodology the origin must be at the center of the man at ground level. This is accomplished by the crouching man offsets

```
float CR_MAN_OFFSET[3] = { 267.6144,123.1392,405.384 } /* in mm      (ft={0.878,0.404,1.33 } */
float ST_MAN_OFFSET[3] = { 267.6144,123.1392,0.0 };    /* in mm      (ft={ 0.878,0.404,0.0 } */
```

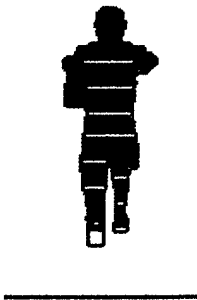


Figure 16 – Anatomy
Floating

Figures 16 and 17 show the position of the man uncorrected by the offsets. The x and y offsets are needed because when viewed from a plan view (Z axis) the origin of the Anatomy files is in the lower left hand corner of the man. This is at the back of the man on his left side. Since the man position is computed by the method in the next section to find the centroid location, the man offsets translate the man to be centered around a

computed point which allows an optimum number of men to fit in a 16 x 16 foot room. This was utilized for consistency with the 2D version which placed 100 men in this size room. Using these offsets 99 men are able to be fit in the standard size room.

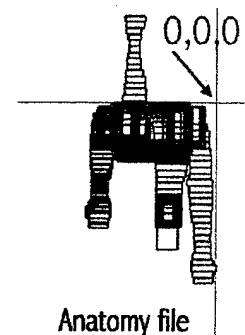


Figure 17 – Anatomy
Origin

Man Positions

As discussed in the previous section, the man positions were calculated to allow as close to 100 men to occupy the standard room. This analysis took into account that the man position is a place for an individual crouching man to be in the room randomly. This means that the room is analyzed as if only one man is in the room at the various man locations. It is these man locations which become incapacitated if a man is in that position in the

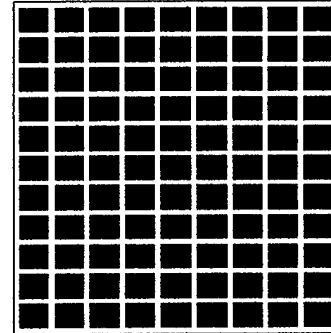


Figure 18 – Man Positions

room. The routine to calculate these values is similar to the 2D version. The code shows that the MAN_DEPTH and the MAN_WIDTH and the num_across and the num_deep drive the output of this routine. In our case, the numbers have been set to match the geometry used to minimize the part of the person outside the typical 16x16 foot room.

```
calc_man_positions(front,side)
float front,side;
{
    int num_across,num_deep,i,j,k;
    float space_across,space_deep,x,y;

    num_across = front / MAN_WIDTH;
    num_deep = side / MAN_DEPTH;
    space_across = (front - (float)num_across * MAN_WIDTH) / 2.0;
    space_deep = (side - (float)num_deep * MAN_DEPTH) / 2.0;

    if((TWOMAN) && (FRONT_WALL/FT2MM > 5) && (SIDE_WALL/FT2MM > 5)){
        twoman[0] =num_across - num_across/2;
        twoman[1] = twoman[0] + num_across +1;}
    else TWOMAN = 0;

    if((SNIPER) && (FRONT_WALL/FT2MM > 5) && (SIDE_WALL/FT2MM > 5)){
        sniper[0] =3 *(num_across - num_across/2);
        sniper[1] = sniper[0] + num_across - 1;
        sniper[2] = sniper[1] + num_across - 1;}
    else SNIPER = 0;

    k=0;
    for (i=0;i<num_deep;i++) {
        y = (side / 2.0) - space_deep - (MAN_DEPTH * (float)i) - MAN_DEPTH / 2.0;
        for (j=0;j<num_across;j++) {
            x = (front / 2.0) - space_across - (MAN_WIDTH * (float)j) - MAN_WIDTH / 2.0;
            men[k].position[0] = x;
            men[k].position[1] = y;
            men[k].position[2] = 0.0;
            ++k; } }
    num_men = k;
}
```

Figure 19 – The Calc man positions Subroutine

DRAFT

Section 3 Correct implementation of wound ballistics model

Pmsincap & Pmssub

This section will address the output result for a fragment or group of fragments that hit one man. For 3dPimms the ComputerMan code developed by ARL is utilized. The code is distributed with modular C++ routines and both a GUI and a batch mode capability. As described in the 3Dpimms users guide, 3dPimms required a new batch mode to allow multiple fragments to be combined before a limb state output was required. A code called pmsincap is a standalone version of the code which is used in 3Dpimms. This code can demonstrate that if a set of fragments and associated hitpoints is input, a specific incapacitation value is output from the ComputerMan model. Also, pmsincap allows a verbose option which allows debugging.

A sample output from pmsincap is shown and the output from a verbose run of 3dp shows that given specific impact conditions the pmsincap & pmssub routines produce the correct output for incapacitation required by 3dp.

Continuous VV&A

This document does not utilize a statistical format for VV&A. Instead graphics based tools have been developed, tested, and utilized to leave with the 3Dpimms method a set of utilities which can help to guaranty ongoing and continuous Verification , Validation and Accreditation activities. This is important because to be useful over time the 3Dpimms methodology must grow with ORCA and the lethality and evaluation community. This section will show in detail the tools which can and have been utilized to answer each of the questions in section 1 – 3.

3dpshow

The first code allows the user to graphically load any fragment dispersion file in a 3dp file. It is based upon all of the methods which are used in 3dp.c . The one exception is that a room full of crouching men is utilized to raytrace instead of the single crouching man BRLCAD file. The features that are available, show which men are hit using raytracing and to report which man number is hit. The graphics code then shows a faceted representation of that man at the appropriate man location. The user is then able to increment the burst point location. The code then displays which men were hit and where in the room they are. The code also displays the shotline or shotlines from the fragment input starting at the appropriate burst point. The code has a toggle for printing out the specific hitpoints in room coordinates. This output can be used as input to pmsincap for V&V. This code was designed to show visually that the proper fragment positions read from the *.3dp file are raytraced from the burst point correctly and produce the correct number of men hit for any burst point. This code, however, does not have the

DRAFT

DRAFT

ability to look at a front or a side shot. The next code in the V&V tools addresses this area.

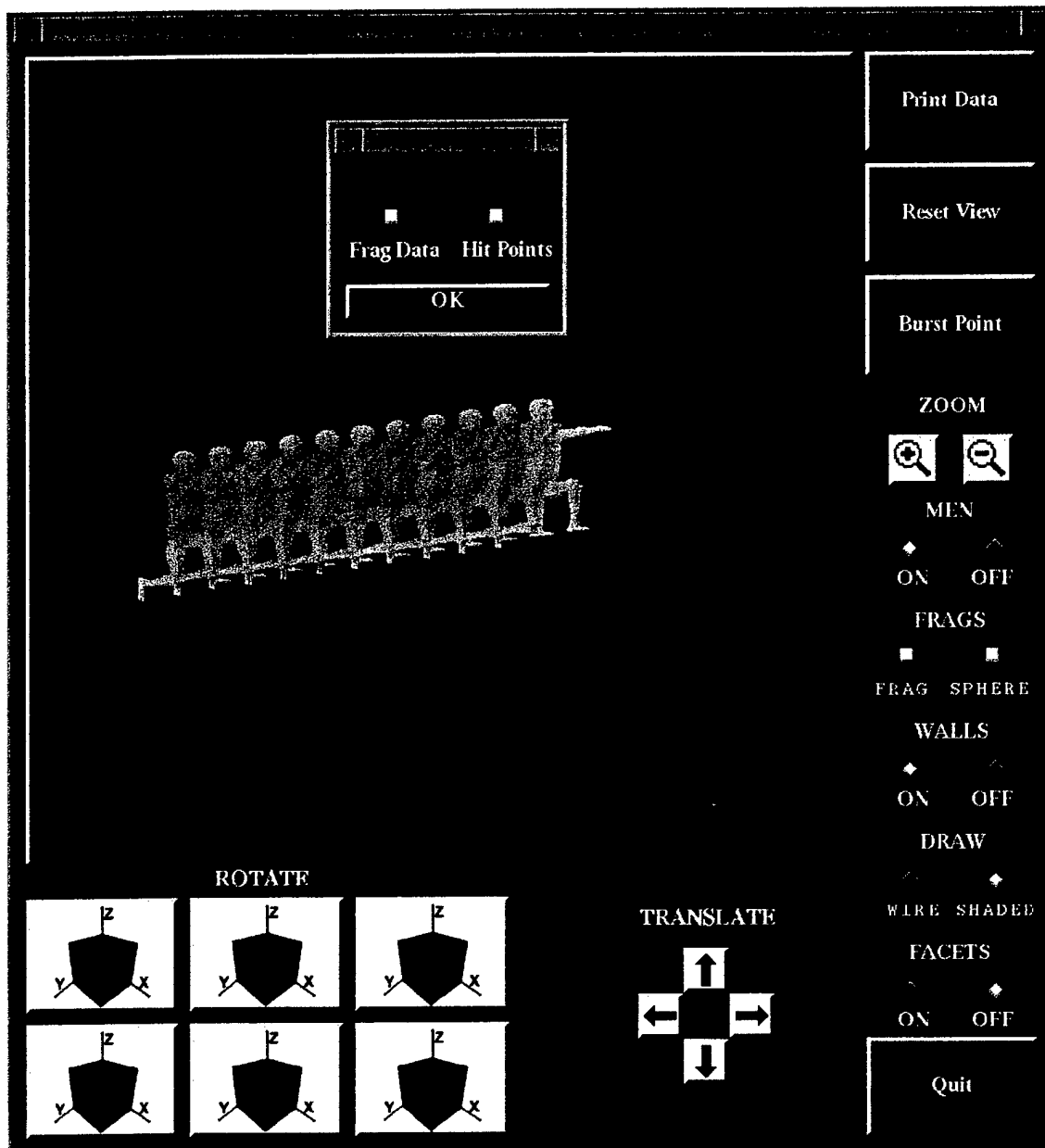


Figure 20 – The 3DPSHOW code and a test file with two fragments hitting in directly opposite directions

Show_V&V

This code allows full control over the man position, the burst point position, the fragment in the 3dp file, and the view. Also, by utilizing the front and side toggle, the shotlines are rotated so that the attack aspect can be modified. This allows the analyst to create special *.3dp files which allow the correct raytrace answers to be calculated by hand and

DRAFT

The check to see if these same results are displayed by the code. One interesting 3dp file has only two fragments at the exactly three feet high and in opposing directions. The entry and exit points are then easily calculated and the expectation is that only at certain burst points will the men in a row be hit. This has been tested and the output of the inhit and outhit point checked. This code also is important to the V&V effort because unlike the previous code, 3dpshow, this code uses only the single man file and the crouching man offset that were discussed in previous sections. Therefore, it the starting point of the ray that is translated relative to the man position and real burst point relative to the crouching man BRL-CAD geometry before the raytrace is accomplished. This code also shows the burst points and the man locations. The man when hit changes color and the display of the hitpoint location is filled with the correct information. An interesting finding during this V&V is that the current definition of front and side are reversed. This does not effect the final answer because both front and side are calculated and averaged separately.

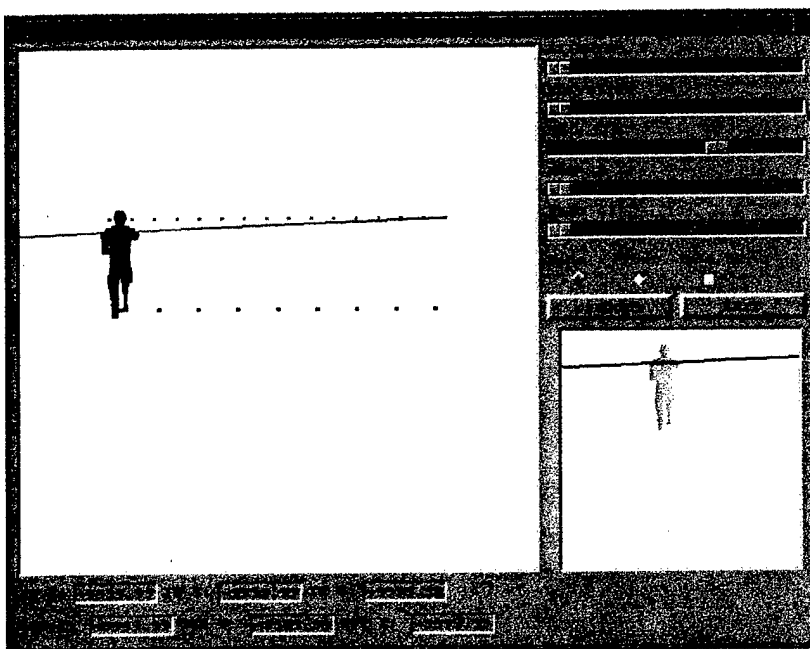
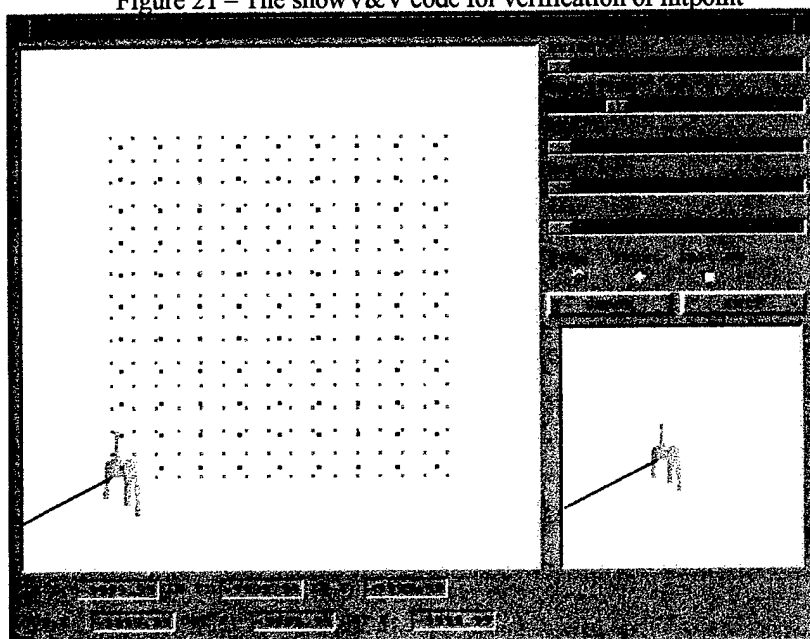


Figure 21 – The showV&V code for verification of hitpoint



DRAFT

Virtual Worlds and Prospect

In an effort to bring the graphics for Verification to an immersive level, a Test Room python script was generated to utilize the Prospect Virtual Reality tool. The concept for use of this tool is that the images from the test can be utilized as texture maps. These texture maps can be applied to representative polygons for each panel. Then bursts can be visualized in each room and the fragment paths will go through the fragment hole in each wall. A bounding Box man has also been converted from the 3dpshow code above and texture mapped with an image. The man can be placed anywhere in the room and the burst file can be read and the burst point set. Rays indicating the path of the fragments are drawn from the burst point extending to the wall. The user can traverse the room by using the mouse to fly until the CTRL and left mouse are hit. The right mouse can be utilized to change the view using it as a virtual trackball. Future plans are to allow the user to verbally call up the burst point, man location, and fragment file. Then issue the fire command and select a fragment. Finally, a raytrace code would indicate which polygons were hit and report in the man's reference system the impact point while calling the incapacitation routine. This would create a fully interactive version of the methodology. Also, the ability to immerse, and fly around the scene yields great advantages for verification and validation. Figure 23 shows a screen with the man and rays.



Figure 23 - The analysis environment in ProspectV2 from Envisage

DRAFT

DRAFT

3DPVV

The main approach for verification of all loops and the logic in the primary code 3dp.c can be found utilizing the output of 3dpvv.c. This code is an exact copy of the current production 3dp.c with a large amount of statements to print out all of the variables and the exact flow of the code. The output from this run can then be checked in depth with the previous visualization and analytical tools. An example of this technique can be seen in this example.

Example of 3dpVV hitpoints checking all of the way to pmsincap run!

Summary

This report is intended to provide support for the Verification decision and to provide enough tools to aide in the Validation effort for this innovative methodology. This report has presented the tools which were built in support of this effort and shown examples of how anyone can verify that the concept model is appropriate and is implemented in the correct manner.

In the process, small improvements have been made. For example, the crouching man was verified by producing a computer code which utilizes ComputerMan routines to create the bounding boxes and to rotate those boxes into the crouching man posture. This model and an earlier hand rotated model were compared for accuracy. The manual model was found to be in excellent agreement with the more precise method. However, since time was available to create a more precise crouching man, this model will be used in the future.

It was also discovered that the definitions of front and side were not implemented across the code. In fact, the numbering of the men and the burst points were rotated from the assumed positions. This is confusing in a way but, the best solution is to leave the numbering and place new labels on the output for the correct side which has been calculated.

APPENDIX III - Geometry Possibilities

BRL-CAD Geometries for Raytrace Linking to ComputerMan / Orca

Glenn Romanczuk

Chris Pitts

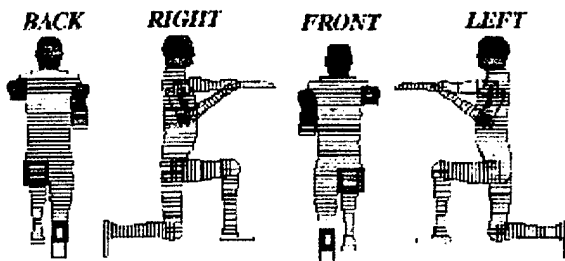
UAH Research Institute
Visualization & Simulation Laboratory
glennr@redstone.army.mil, cmpitts@redstone.army.mil

Introduction

UAH Research Institute identified possible problems in the calculation of Probability of Incapacitation utilizing the Pimms/Picode methods. This early work led to the concept of utilizing ComputerMan as a methodology for determining incapacitation for the MPIM program. With MPIM funding UAH RI created several different concepts of ways to use raytracing in BRL-CAD to replace the statistical assumptions in Pimms. This paper will explore some of this early work and also the creation of files which properly allow linkage to ComputerMan / ORCA.

Approach

The first approaches that were attempted involved using a specific tissue index from the ComputerMan data to gather the necessary geometric information required to build a BRLCAD file with the appropriate scale and surface position. Tools which are included with the ComputeMan system were used to extract the tissue positions for all skin cells. These skin cell positions are cell index points for a specific section in the ComputerMan anatomy file. This data was then utilized to build a box or ARB8 representing each skin cell. It was thought that this geometry would be usable to define all hits on the man in the right coordinate space. Figure 1 shows the result rendered in using rt. Although this geometry has the main outline of the standing man geometry several areas can be seen which make this approach suboptimal. In the feet, it is clear that a ray could pass through this geometry. Also, the size of the file created was rather cumbersome. The third problem with this approach was the ability to rotate the cells into the other positions required for use in linking to ComputerMan. With these problems in mind another approach was attempted using the method utilized by the GUI.



Bounding Box Method

Figure 1- The skin section .g file
A quick look at the coded version of the information used by the Graphical User Interface (GUI) suggested the next method. This method was to utilize the bounding box data for every section at a specific z level and to construct a chunky

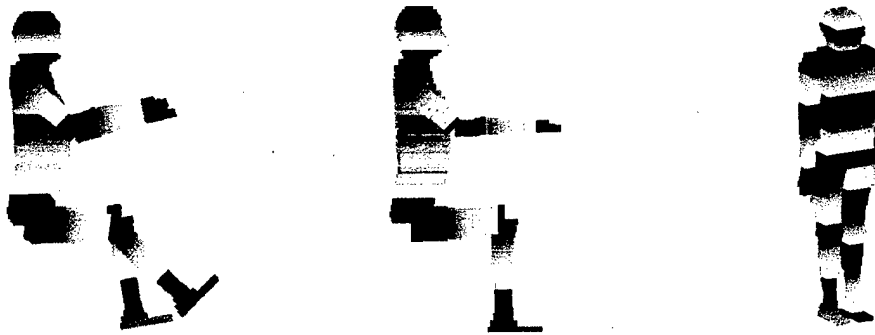
Figure 2 – The Crouching man saved from GUI

representation of the Anatomy in the proper coordinate frame. Figure 2 shows the crouching man in the ComputerMan GUI. The GUI utilizes the bounding box visual to orient the wound path. Therefore, the bounding box data was extracted from the ComputerMan source code and a program was built to create an input script to import into BRLCAD and create a BoundingBox file in mged *.g format. This led to the creation of the bbman.g file which is rendered in Figure 3. This figure shows the man transformed into the crouching posture by rotation of certain bounding box regions. This process was at first manually accomplished but now a method is utilized straight from the ComputerMan reference documentation to rotate the bounding boxes by the correct matrix. Figures 4-6 show the other postures that have been developed from the rotation angles stored in the ComputerMan GUI.



Figure 3 – oblique view of crouching man .g file.

Current Efforts



Figures 4,5,6 The Driving, Sitting, and Standing Bounding Box men shown faceted from .g file.

One criticism of the Bounding Box man is that he is chunky and a raytrace will not provide a correct obliquity value for ricochet calculation or for other algorithms for velocity degradation due to clothing or armor. These are valid limitations of using Bounding Boxes as geometry to raytrace for linking to ComputerMan/ORCA. The suggestion has been made to utilize another solid rather than a box or ARB8. The TEC or TGC solid appear to be the right type of solid to add resolution but to keep all of the benefits of this approach.

Figure 7 shows a quick implementation of using tgc's as the element. This method should be checked at each section level to compare with the skin cells to make sure that this is the minimum enclosing tgc to fit the skin cells. Also, a check should be made to be sure that the tgc does not cut too deeply through into the volume occupied by the skin cells. This is a first cut because we did not try to constrain the tgc's to have the same size at the top and bottom of adjoining cell. This method would produce a smoother surface, although it would add to the areas where the tgc model does not have similar tissue in the database of the Anatomy file.



Figure 7 – A standing Tgc man in Brl-Cad and in a faceted form converted using Fred.

Figure 8 & 9 show two different results when utilizing the bounding box data to generate tgc's at each section level. The first shows the proper results which encase the skin cells in a tgc. The second figure shows a problem are with the quick approach. Therefore, the analyst must find the necessary values for input to the tgc at this section level.

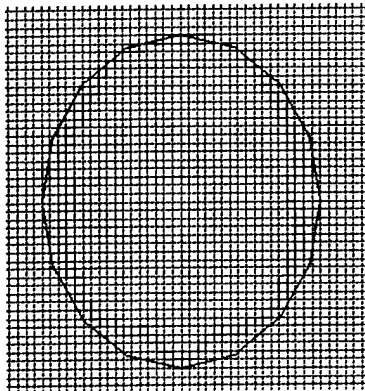


Figure 8 – Section 10 of skin guy with tgc shown

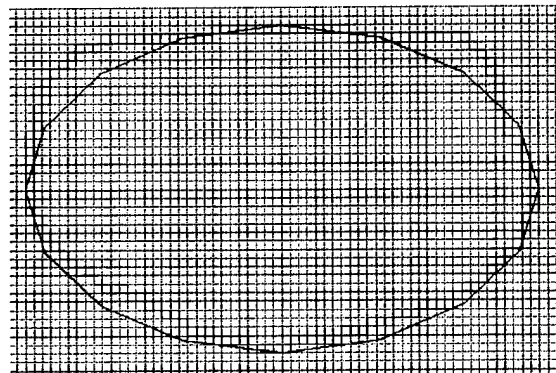
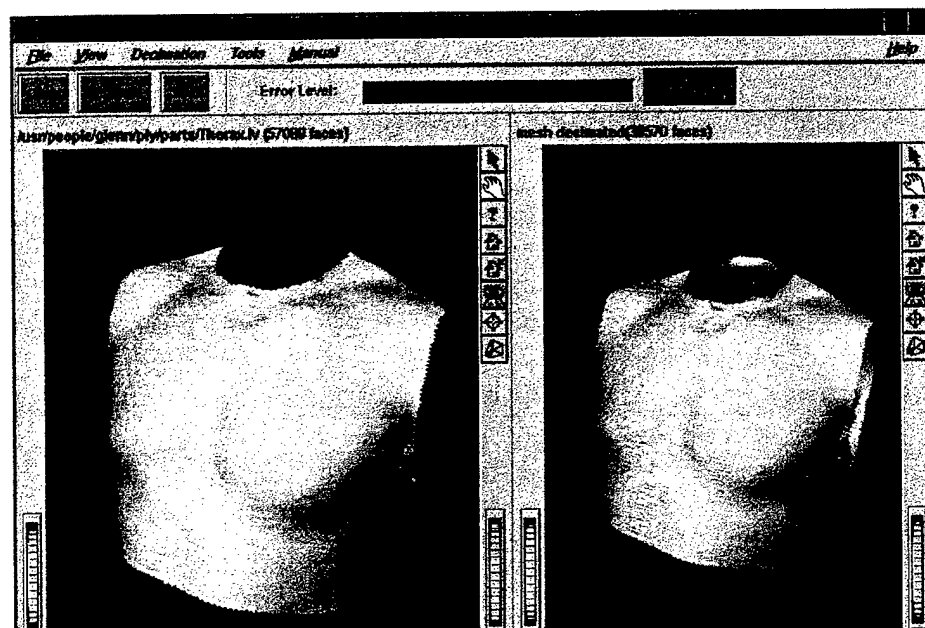
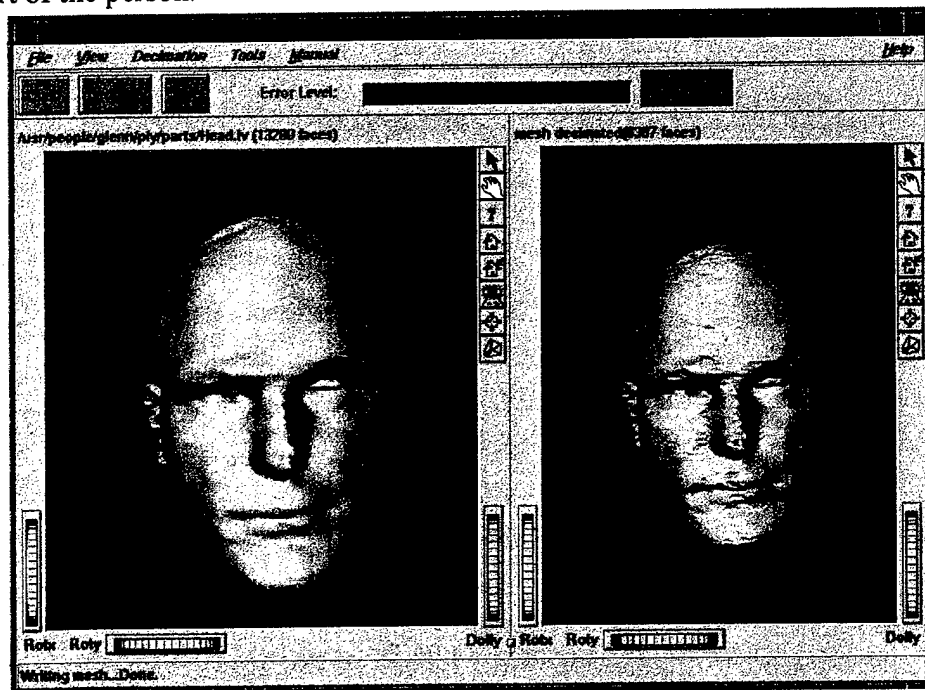


Figure 9 – Section 40 showing skin cells outside the tgc autocreated using the bounding box parameters

Other Efforts and Discussion

Decimation is one method of polygon reduction. This is a rapidly changing field which is of particular interest to analysts involved in real-time simulation, animation experts, and people with large models. Laser range scanners which can produce detailed polygonal models of the human body and other shapes also typically output large models. Several methods have been examined to reduce the test model in the ply format to other usable formats. There are a number of free decimation tools available on the web. The two that have been examined are the Qvis/Qslim and JadeV21. The test ply object was converted to a wavefront *.obj file using appropriate conversion tools. Also, the model was reduced to separate parts files for the arms, legs, thorax, etc.. Having the decimation technique apply to smaller sub models should allow for different error tolerance values to be set for each part of the person.



These two examples show that significant reduction in the number of facets can be accomplished without reduction in the overall shape and with a limited error from the exact surface. These examples lead to a proposed solution for using the range scan data in the linkage to ORCA. If a range scan person could be put in the appropriate poses etc.. it would be possible to input, divide into parts, decimate, then import in to BRL-CAD using the ARB6 as the solid of choice, these parts would then be made into regions. Therefore, BRL-CAD raytracing could be used on this high resolution person in conjunction with the downsampled tgc man. By having both representation in BRL-CAD boolean operations could be utilized looking at differences. Also, once in BRL-CAD automatic uparmoring could be conducted using Libwdb or other automated techniques which already exist in the lethality/survivability world. This also would help the extensibility of the approaches used in this simulation to be utilized for other purposes.

Conclusions

Several methods of utilizing data in the ComputerMan / Orca distribution and having there basis in the underlying Anatomy have been shown. These methods on the low fidelity end utilize only the bounding box information and the rotation and translation information for the various postures. A more robust representation can be accomplished utilizing skin cells extracted from the Anatomy if areas of error can be accepted. However, the tgc method if created with caution, leads to the most accurate BRL-CAD representation of the ComputerMan/Orca Anatomy description for raytrace. Similar methods to our creatman.c code could be used on the tgc man to re-create the appropriate postures.

The two millimeter resolution ply files testman.ply has been reduced into appropriate parts, decimated into a sample to see the effects and converted into a ARB6 representation for raytrace in BRL-CAD. These results of this work provide a path forward for both the efforts of Mr. Rosenblatt and ARL in general for incorporation of ComputerMan/Orca in vehicles (like the BMP3) where munitions have an incapacitation requirement. These computations might be more CPU intensive than the current Sperazza/Kokinakis or Ballistic Dose computations, however, this will ensure that across the application areas for incapacitation calculation there will not be apples compared to oranges.